

lpopt
trunk

Generated by Doxygen 1.8.5

Mon Oct 21 2013 19:07:48

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	2
2.1	Class Hierarchy	2
3	Class Index	15
3.1	Class List	15
4	File Index	26
4.1	File List	26
5	Namespace Documentation	32
5.1	Ipopt Namespace Reference	32
5.1.1	Typedef Documentation	44
5.1.2	Enumeration Type Documentation	45
5.1.3	Function Documentation	49
6	Class Documentation	55
6.1	Ipopt::AdaptiveMuUpdate Class Reference	55
6.1.1	Detailed Description	58
6.1.2	Member Enumeration Documentation	58
6.1.3	Constructor & Destructor Documentation	59
6.1.4	Member Function Documentation	59
6.1.5	Member Data Documentation	60
6.2	Ipopt::AlgorithmBuilder Class Reference	63
6.2.1	Detailed Description	64
6.2.2	Constructor & Destructor Documentation	64
6.2.3	Member Function Documentation	65
6.2.4	Member Data Documentation	65
6.3	Ipopt::AlgorithmStrategyObject Class Reference	65
6.3.1	Detailed Description	67
6.3.2	Constructor & Destructor Documentation	67
6.3.3	Member Function Documentation	68
6.3.4	Member Data Documentation	69
6.4	Ipopt::AmplOptionsList::AmplOption Class Reference	70
6.4.1	Detailed Description	70

6.4.2	Constructor & Destructor Documentation	71
6.4.3	Member Function Documentation	71
6.4.4	Member Data Documentation	71
6.5	Ipopt::AmplOptionsList Class Reference	72
6.5.1	Detailed Description	73
6.5.2	Member Enumeration Documentation	73
6.5.3	Constructor & Destructor Documentation	73
6.5.4	Member Function Documentation	74
6.5.5	Member Data Documentation	74
6.6	Ipopt::AmplSuffixHandler Class Reference	74
6.6.1	Detailed Description	76
6.6.2	Member Enumeration Documentation	76
6.6.3	Constructor & Destructor Documentation	76
6.6.4	Member Function Documentation	76
6.6.5	Friends And Related Function Documentation	77
6.6.6	Member Data Documentation	77
6.7	Ipopt::AmplTNLP Class Reference	77
6.7.1	Detailed Description	81
6.7.2	Constructor & Destructor Documentation	81
6.7.3	Member Function Documentation	81
6.7.4	Member Data Documentation	85
6.8	Ipopt::AugRestoSystemSolver Class Reference	87
6.8.1	Detailed Description	89
6.8.2	Constructor & Destructor Documentation	89
6.8.3	Member Function Documentation	90
6.8.4	Member Data Documentation	91
6.9	Ipopt::AugSystemSolver Class Reference	92
6.9.1	Detailed Description	94
6.9.2	Constructor & Destructor Documentation	94
6.9.3	Member Function Documentation	94
6.10	Ipopt::BacktrackingLineSearch Class Reference	96
6.10.1	Detailed Description	99
6.10.2	Member Enumeration Documentation	100
6.10.3	Constructor & Destructor Documentation	100
6.10.4	Member Function Documentation	100
6.10.5	Member Data Documentation	103
6.11	Ipopt::BacktrackingLSAcceptor Class Reference	107

6.11.1 Detailed Description	108
6.11.2 Constructor & Destructor Documentation	108
6.11.3 Member Function Documentation	109
6.12 Ipopt::CachedResults< T > Class Template Reference	112
6.12.1 Detailed Description	113
6.12.2 Constructor & Destructor Documentation	114
6.12.3 Member Function Documentation	114
6.12.4 Member Data Documentation	117
6.13 Ipopt::CGPenaltyCq Class Reference	117
6.13.1 Detailed Description	119
6.13.2 Constructor & Destructor Documentation	119
6.13.3 Member Function Documentation	119
6.13.4 Member Data Documentation	121
6.14 Ipopt::CGPenaltyData Class Reference	122
6.14.1 Detailed Description	124
6.14.2 Constructor & Destructor Documentation	124
6.14.3 Member Function Documentation	125
6.14.4 Member Data Documentation	127
6.15 Ipopt::CGPenaltyLSAcceptor Class Reference	128
6.15.1 Detailed Description	132
6.15.2 Constructor & Destructor Documentation	132
6.15.3 Member Function Documentation	132
6.15.4 Member Data Documentation	135
6.16 Ipopt::CGPerturbationHandler Class Reference	139
6.16.1 Detailed Description	141
6.16.2 Member Enumeration Documentation	142
6.16.3 Constructor & Destructor Documentation	142
6.16.4 Member Function Documentation	142
6.16.5 Member Data Documentation	144
6.17 Ipopt::CGSearchDirCalculator Class Reference	146
6.17.1 Detailed Description	148
6.17.2 Constructor & Destructor Documentation	148
6.17.3 Member Function Documentation	149
6.17.4 Member Data Documentation	149
6.18 Ipopt::CompoundMatrix Class Reference	151
6.18.1 Detailed Description	153
6.18.2 Constructor & Destructor Documentation	153

6.18.3	Member Function Documentation	153
6.18.4	Member Data Documentation	155
6.19	Ipopt::CompoundMatrixSpace Class Reference	156
6.19.1	Detailed Description	158
6.19.2	Constructor & Destructor Documentation	158
6.19.3	Member Function Documentation	158
6.19.4	Member Data Documentation	159
6.20	Ipopt::CompoundSymMatrix Class Reference	160
6.20.1	Detailed Description	162
6.20.2	Constructor & Destructor Documentation	162
6.20.3	Member Function Documentation	163
6.20.4	Member Data Documentation	164
6.21	Ipopt::CompoundSymMatrixSpace Class Reference	165
6.21.1	Detailed Description	166
6.21.2	Constructor & Destructor Documentation	167
6.21.3	Member Function Documentation	167
6.21.4	Member Data Documentation	168
6.22	Ipopt::CompoundVector Class Reference	168
6.22.1	Detailed Description	171
6.22.2	Constructor & Destructor Documentation	171
6.22.3	Member Function Documentation	172
6.22.4	Member Data Documentation	175
6.23	Ipopt::CompoundVectorSpace Class Reference	176
6.23.1	Detailed Description	177
6.23.2	Constructor & Destructor Documentation	177
6.23.3	Member Function Documentation	178
6.23.4	Member Data Documentation	179
6.24	Ipopt::ConvergenceCheck Class Reference	180
6.24.1	Detailed Description	181
6.24.2	Member Enumeration Documentation	181
6.24.3	Constructor & Destructor Documentation	181
6.24.4	Member Function Documentation	182
6.25	Ipopt::DefaultIterateInitializer Class Reference	182
6.25.1	Detailed Description	184
6.25.2	Member Enumeration Documentation	185
6.25.3	Constructor & Destructor Documentation	185
6.25.4	Member Function Documentation	185

6.25.5	Member Data Documentation	186
6.26	Ipopt::DenseGenMatrix Class Reference	188
6.26.1	Detailed Description	190
6.26.2	Member Enumeration Documentation	190
6.26.3	Constructor & Destructor Documentation	190
6.26.4	Member Function Documentation	191
6.26.5	Member Data Documentation	193
6.27	Ipopt::DenseGenMatrixSpace Class Reference	194
6.27.1	Detailed Description	194
6.27.2	Constructor & Destructor Documentation	195
6.27.3	Member Function Documentation	195
6.28	Ipopt::DenseSymMatrix Class Reference	195
6.28.1	Detailed Description	197
6.28.2	Constructor & Destructor Documentation	197
6.28.3	Member Function Documentation	197
6.28.4	Member Data Documentation	199
6.29	Ipopt::DenseSymMatrixSpace Class Reference	199
6.29.1	Detailed Description	200
6.29.2	Constructor & Destructor Documentation	200
6.29.3	Member Function Documentation	200
6.30	Ipopt::DenseVector Class Reference	200
6.30.1	Detailed Description	204
6.30.2	Constructor & Destructor Documentation	204
6.30.3	Member Function Documentation	204
6.30.4	Friends And Related Function Documentation	208
6.30.5	Member Data Documentation	208
6.31	Ipopt::DenseVectorSpace Class Reference	209
6.31.1	Detailed Description	210
6.31.2	Constructor & Destructor Documentation	211
6.31.3	Member Function Documentation	211
6.31.4	Member Data Documentation	212
6.32	Ipopt::DependentResult< T > Class Template Reference	213
6.32.1	Detailed Description	214
6.32.2	Constructor & Destructor Documentation	214
6.32.3	Member Function Documentation	215
6.32.4	Member Data Documentation	216
6.33	Ipopt::DiagMatrix Class Reference	216

6.33.1 Detailed Description	218
6.33.2 Constructor & Destructor Documentation	218
6.33.3 Member Function Documentation	218
6.33.4 Member Data Documentation	219
6.34 Ipopt::DiagMatrixSpace Class Reference	219
6.34.1 Detailed Description	220
6.34.2 Constructor & Destructor Documentation	220
6.34.3 Member Function Documentation	221
6.35 Ipopt::EqMultiplierCalculator Class Reference	221
6.35.1 Detailed Description	222
6.35.2 Constructor & Destructor Documentation	222
6.35.3 Member Function Documentation	223
6.36 Ipopt::EquilibrationScaling Class Reference	223
6.36.1 Detailed Description	224
6.36.2 Constructor & Destructor Documentation	224
6.36.3 Member Function Documentation	225
6.36.4 Member Data Documentation	225
6.37 Ipopt::ExactHessianUpdater Class Reference	226
6.37.1 Detailed Description	227
6.37.2 Constructor & Destructor Documentation	227
6.37.3 Member Function Documentation	227
6.38 Ipopt::ExpandedMultiVectorMatrix Class Reference	227
6.38.1 Detailed Description	229
6.38.2 Constructor & Destructor Documentation	229
6.38.3 Member Function Documentation	230
6.38.4 Member Data Documentation	231
6.39 Ipopt::ExpandedMultiVectorMatrixSpace Class Reference	231
6.39.1 Detailed Description	232
6.39.2 Constructor & Destructor Documentation	232
6.39.3 Member Function Documentation	233
6.39.4 Member Data Documentation	233
6.40 Ipopt::ExpansionMatrix Class Reference	233
6.40.1 Detailed Description	235
6.40.2 Constructor & Destructor Documentation	235
6.40.3 Member Function Documentation	236
6.40.4 Friends And Related Function Documentation	237
6.40.5 Member Data Documentation	237

6.41	Ipopt::ExpansionMatrixSpace Class Reference	237
6.41.1	Detailed Description	238
6.41.2	Constructor & Destructor Documentation	238
6.41.3	Member Function Documentation	239
6.41.4	Member Data Documentation	239
6.42	Ipopt::FileJournal Class Reference	240
6.42.1	Detailed Description	241
6.42.2	Constructor & Destructor Documentation	241
6.42.3	Member Function Documentation	241
6.42.4	Member Data Documentation	242
6.43	Ipopt::Filter Class Reference	242
6.43.1	Detailed Description	243
6.43.2	Constructor & Destructor Documentation	243
6.43.3	Member Function Documentation	243
6.43.4	Member Data Documentation	244
6.44	Ipopt::FilterEntry Class Reference	244
6.44.1	Detailed Description	245
6.44.2	Constructor & Destructor Documentation	245
6.44.3	Member Function Documentation	246
6.44.4	Member Data Documentation	246
6.45	Ipopt::FilterLSAcceptor Class Reference	247
6.45.1	Detailed Description	250
6.45.2	Member Enumeration Documentation	250
6.45.3	Constructor & Destructor Documentation	250
6.45.4	Member Function Documentation	251
6.45.5	Member Data Documentation	253
6.46	Ipopt::GenAugSystemSolver Class Reference	256
6.46.1	Detailed Description	258
6.46.2	Constructor & Destructor Documentation	258
6.46.3	Member Function Documentation	259
6.46.4	Member Data Documentation	260
6.47	Ipopt::GenKKTsSolverInterface Class Reference	262
6.47.1	Detailed Description	262
6.47.2	Constructor & Destructor Documentation	262
6.47.3	Member Function Documentation	263
6.48	Ipopt::GenTMatrix Class Reference	264
6.48.1	Detailed Description	266

6.48.2	Constructor & Destructor Documentation	266
6.48.3	Member Function Documentation	267
6.48.4	Friends And Related Function Documentation	268
6.48.5	Member Data Documentation	268
6.49	Ipopt::GenTMatrixSpace Class Reference	269
6.49.1	Detailed Description	270
6.49.2	Constructor & Destructor Documentation	270
6.49.3	Member Function Documentation	271
6.49.4	Friends And Related Function Documentation	271
6.49.5	Member Data Documentation	271
6.50	Ipopt::GradientScaling Class Reference	272
6.50.1	Detailed Description	273
6.50.2	Constructor & Destructor Documentation	273
6.50.3	Member Function Documentation	274
6.50.4	Member Data Documentation	274
6.51	Ipopt::HessianUpdater Class Reference	275
6.51.1	Detailed Description	276
6.51.2	Constructor & Destructor Documentation	276
6.51.3	Member Function Documentation	276
6.52	Ipopt::IdentityMatrix Class Reference	276
6.52.1	Detailed Description	278
6.52.2	Constructor & Destructor Documentation	278
6.52.3	Member Function Documentation	278
6.52.4	Member Data Documentation	279
6.53	Ipopt::IdentityMatrixSpace Class Reference	280
6.53.1	Detailed Description	281
6.53.2	Constructor & Destructor Documentation	281
6.53.3	Member Function Documentation	281
6.54	Ipopt::InexactAlgorithmBuilder Class Reference	281
6.54.1	Detailed Description	283
6.54.2	Constructor & Destructor Documentation	283
6.54.3	Member Function Documentation	283
6.54.4	Member Data Documentation	283
6.55	Ipopt::InexactCq Class Reference	284
6.55.1	Detailed Description	286
6.55.2	Constructor & Destructor Documentation	286
6.55.3	Member Function Documentation	286

6.55.4	Member Data Documentation	288
6.56	Ipopt::InexactData Class Reference	289
6.56.1	Detailed Description	291
6.56.2	Constructor & Destructor Documentation	291
6.56.3	Member Function Documentation	291
6.56.4	Member Data Documentation	293
6.57	Ipopt::InexactDoglegNormalStep Class Reference	294
6.57.1	Detailed Description	295
6.57.2	Constructor & Destructor Documentation	295
6.57.3	Member Function Documentation	296
6.57.4	Member Data Documentation	296
6.58	Ipopt::InexactLSAcceptor Class Reference	297
6.58.1	Detailed Description	300
6.58.2	Constructor & Destructor Documentation	300
6.58.3	Member Function Documentation	300
6.58.4	Member Data Documentation	302
6.59	Ipopt::InexactNewtonNormalStep Class Reference	305
6.59.1	Detailed Description	306
6.59.2	Constructor & Destructor Documentation	306
6.59.3	Member Function Documentation	307
6.59.4	Member Data Documentation	307
6.60	Ipopt::InexactNormalStepCalculator Class Reference	308
6.60.1	Detailed Description	309
6.60.2	Constructor & Destructor Documentation	309
6.60.3	Member Function Documentation	309
6.61	Ipopt::InexactNormalTerminationTester Class Reference	310
6.61.1	Detailed Description	311
6.61.2	Constructor & Destructor Documentation	311
6.61.3	Member Function Documentation	312
6.61.4	Member Data Documentation	313
6.62	Ipopt::InexactPDSolver Class Reference	313
6.62.1	Detailed Description	315
6.62.2	Constructor & Destructor Documentation	315
6.62.3	Member Function Documentation	315
6.62.4	Member Data Documentation	316
6.63	Ipopt::InexactPDTerminationTester Class Reference	317
6.63.1	Detailed Description	319

6.63.2	Constructor & Destructor Documentation	319
6.63.3	Member Function Documentation	320
6.63.4	Member Data Documentation	320
6.64	Ipopt::InexactSearchDirCalculator Class Reference	323
6.64.1	Detailed Description	325
6.64.2	Member Enumeration Documentation	325
6.64.3	Constructor & Destructor Documentation	325
6.64.4	Member Function Documentation	326
6.64.5	Member Data Documentation	326
6.65	Ipopt::InexactTSymScalingMethod Class Reference	327
6.65.1	Detailed Description	328
6.65.2	Constructor & Destructor Documentation	328
6.65.3	Member Function Documentation	328
6.66	Ipopt::IpoptAdditionalCq Class Reference	329
6.66.1	Detailed Description	329
6.66.2	Constructor & Destructor Documentation	330
6.66.3	Member Function Documentation	330
6.67	Ipopt::IpoptAdditionalData Class Reference	330
6.67.1	Detailed Description	331
6.67.2	Constructor & Destructor Documentation	331
6.67.3	Member Function Documentation	332
6.68	Ipopt::IpoptAlgorithm Class Reference	332
6.68.1	Detailed Description	335
6.68.2	Constructor & Destructor Documentation	335
6.68.3	Member Function Documentation	335
6.68.4	Member Data Documentation	337
6.69	Ipopt::IpoptApplication Class Reference	338
6.69.1	Detailed Description	341
6.69.2	Constructor & Destructor Documentation	341
6.69.3	Member Function Documentation	341
6.69.4	Member Data Documentation	344
6.70	Ipopt::IpoptCalculatedQuantities Class Reference	345
6.70.1	Detailed Description	355
6.70.2	Constructor & Destructor Documentation	355
6.70.3	Member Function Documentation	355
6.70.4	Member Data Documentation	365
6.71	Ipopt::IpoptData Class Reference	374

6.71.1 Detailed Description	379
6.71.2 Constructor & Destructor Documentation	379
6.71.3 Member Function Documentation	380
6.71.4 Member Data Documentation	386
6.72 Ipopt::IpoptException Class Reference	389
6.72.1 Detailed Description	390
6.72.2 Constructor & Destructor Documentation	390
6.72.3 Member Function Documentation	391
6.72.4 Member Data Documentation	391
6.73 Ipopt::IpoptNLP Class Reference	391
6.73.1 Detailed Description	394
6.73.2 Constructor & Destructor Documentation	394
6.73.3 Member Function Documentation	394
6.73.4 Member Data Documentation	398
6.74 Ipopt::IterateInitializer Class Reference	399
6.74.1 Detailed Description	400
6.74.2 Constructor & Destructor Documentation	400
6.74.3 Member Function Documentation	400
6.75 Ipopt::IteratesVector Class Reference	400
6.75.1 Detailed Description	404
6.75.2 Constructor & Destructor Documentation	404
6.75.3 Member Function Documentation	405
6.75.4 Member Data Documentation	411
6.76 Ipopt::IteratesVectorSpace Class Reference	412
6.76.1 Detailed Description	413
6.76.2 Constructor & Destructor Documentation	413
6.76.3 Member Function Documentation	413
6.76.4 Member Data Documentation	414
6.77 Ipopt::IterationOutput Class Reference	415
6.77.1 Detailed Description	416
6.77.2 Member Enumeration Documentation	416
6.77.3 Constructor & Destructor Documentation	416
6.77.4 Member Function Documentation	417
6.78 Ipopt::IterativePardisoSolverInterface Class Reference	417
6.78.1 Detailed Description	421
6.78.2 Member Enumeration Documentation	421
6.78.3 Constructor & Destructor Documentation	421

6.78.4	Member Function Documentation	422
6.78.5	Member Data Documentation	423
6.79	Ipopt::IterativeSolverTerminationTester Class Reference	427
6.79.1	Detailed Description	428
6.79.2	Member Enumeration Documentation	429
6.79.3	Constructor & Destructor Documentation	429
6.79.4	Member Function Documentation	429
6.80	Ipopt::IterativeWsmSolverInterface Class Reference	430
6.80.1	Detailed Description	433
6.80.2	Constructor & Destructor Documentation	433
6.80.3	Member Function Documentation	433
6.80.4	Member Data Documentation	435
6.81	Ipopt::Journal Class Reference	436
6.81.1	Detailed Description	438
6.81.2	Constructor & Destructor Documentation	438
6.81.3	Member Function Documentation	438
6.81.4	Member Data Documentation	439
6.82	Ipopt::Journalist Class Reference	440
6.82.1	Detailed Description	441
6.82.2	Constructor & Destructor Documentation	442
6.82.3	Member Function Documentation	442
6.82.4	Member Data Documentation	443
6.83	Ipopt::LeastSquareMultipliers Class Reference	444
6.83.1	Detailed Description	445
6.83.2	Constructor & Destructor Documentation	445
6.83.3	Member Function Documentation	445
6.83.4	Member Data Documentation	445
6.84	Ipopt::LimMemQuasiNewtonUpdater Class Reference	446
6.84.1	Detailed Description	450
6.84.2	Member Enumeration Documentation	450
6.84.3	Constructor & Destructor Documentation	451
6.84.4	Member Function Documentation	451
6.84.5	Member Data Documentation	454
6.85	Ipopt::LineSearch Class Reference	459
6.85.1	Detailed Description	460
6.85.2	Constructor & Destructor Documentation	460
6.85.3	Member Function Documentation	460

6.86	Ipopt::LogMuOracle Class Reference	461
6.86.1	Detailed Description	462
6.86.2	Constructor & Destructor Documentation	462
6.86.3	Member Function Documentation	463
6.87	Ipopt::LowRankAugSystemSolver Class Reference	463
6.87.1	Detailed Description	466
6.87.2	Constructor & Destructor Documentation	466
6.87.3	Member Function Documentation	466
6.87.4	Member Data Documentation	467
6.88	Ipopt::LowRankSSAugSystemSolver Class Reference	470
6.88.1	Detailed Description	472
6.88.2	Constructor & Destructor Documentation	472
6.88.3	Member Function Documentation	473
6.88.4	Member Data Documentation	474
6.89	Ipopt::LowRankUpdateSymMatrix Class Reference	476
6.89.1	Detailed Description	478
6.89.2	Constructor & Destructor Documentation	478
6.89.3	Member Function Documentation	479
6.89.4	Member Data Documentation	480
6.90	Ipopt::LowRankUpdateSymMatrixSpace Class Reference	481
6.90.1	Detailed Description	482
6.90.2	Constructor & Destructor Documentation	482
6.90.3	Member Function Documentation	483
6.90.4	Member Data Documentation	483
6.91	Ipopt::Ma27TSolverInterface Class Reference	484
6.91.1	Detailed Description	487
6.91.2	Constructor & Destructor Documentation	487
6.91.3	Member Function Documentation	487
6.91.4	Member Data Documentation	489
6.92	Ipopt::Ma28TDependencyDetector Class Reference	491
6.92.1	Detailed Description	493
6.92.2	Constructor & Destructor Documentation	493
6.92.3	Member Function Documentation	493
6.92.4	Member Data Documentation	493
6.93	Ipopt::Ma57TSolverInterface Class Reference	494
6.93.1	Detailed Description	496
6.93.2	Constructor & Destructor Documentation	496

6.93.3 Member Function Documentation	497
6.93.4 Member Data Documentation	498
6.94 ma77_control_d Struct Reference	500
6.94.1 Detailed Description	501
6.94.2 Member Data Documentation	501
6.95 ma77_info_d Struct Reference	503
6.95.1 Detailed Description	504
6.95.2 Member Data Documentation	504
6.96 Ipopt::Ma77SolverInterface Class Reference	507
6.96.1 Detailed Description	508
6.96.2 Member Enumeration Documentation	509
6.96.3 Constructor & Destructor Documentation	509
6.96.4 Member Function Documentation	509
6.96.5 Member Data Documentation	511
6.97 ma86_control_d Struct Reference	512
6.97.1 Detailed Description	512
6.97.2 Member Data Documentation	512
6.98 ma86_info_d Struct Reference	514
6.98.1 Detailed Description	514
6.98.2 Member Data Documentation	514
6.99 Ipopt::Ma86SolverInterface Class Reference	515
6.99.1 Detailed Description	517
6.99.2 Member Enumeration Documentation	518
6.99.3 Constructor & Destructor Documentation	518
6.99.4 Member Function Documentation	518
6.99.5 Member Data Documentation	520
6.100ma97_control_d Struct Reference	521
6.100.1 Detailed Description	521
6.100.2 Member Data Documentation	521
6.101ma97_info Struct Reference	523
6.101.1 Detailed Description	523
6.101.2 Member Data Documentation	523
6.102Ipopt::Ma97SolverInterface Class Reference	525
6.102.1 Detailed Description	527
6.102.2 Member Enumeration Documentation	528
6.102.3 Constructor & Destructor Documentation	528
6.102.4 Member Function Documentation	529

6.102.5 Member Data Documentation	530
6.103Ipopt::Matrix Class Reference	532
6.103.1 Detailed Description	535
6.103.2 Constructor & Destructor Documentation	535
6.103.3 Member Function Documentation	536
6.103.4 Member Data Documentation	539
6.104Ipopt::MatrixSpace Class Reference	539
6.104.1 Detailed Description	540
6.104.2 Constructor & Destructor Documentation	541
6.104.3 Member Function Documentation	541
6.104.4 Member Data Documentation	541
6.105Ipopt::Mc19TSymScalingMethod Class Reference	542
6.105.1 Detailed Description	543
6.105.2 Constructor & Destructor Documentation	543
6.105.3 Member Function Documentation	543
6.106mc68_control Struct Reference	543
6.106.1 Detailed Description	544
6.106.2 Member Data Documentation	544
6.107mc68_info Struct Reference	545
6.107.1 Detailed Description	545
6.107.2 Member Data Documentation	545
6.108Ipopt::MinC_1NrmRestorationPhase Class Reference	546
6.108.1 Detailed Description	548
6.108.2 Constructor & Destructor Documentation	548
6.108.3 Member Function Documentation	548
6.108.4 Member Data Documentation	549
6.109Ipopt::MonotoneMuUpdate Class Reference	550
6.109.1 Detailed Description	551
6.109.2 Constructor & Destructor Documentation	551
6.109.3 Member Function Documentation	552
6.109.4 Member Data Documentation	552
6.110Ipopt::MultiVectorMatrix Class Reference	553
6.110.1 Detailed Description	555
6.110.2 Constructor & Destructor Documentation	555
6.110.3 Member Function Documentation	556
6.110.4 Member Data Documentation	558
6.111Ipopt::MultiVectorMatrixSpace Class Reference	558

6.111.1 Detailed Description	559
6.111.2 Constructor & Destructor Documentation	559
6.111.3 Member Function Documentation	559
6.111.4 Member Data Documentation	560
6.112Ipopt::MumpsSolverInterface Class Reference	560
6.112.1 Detailed Description	562
6.112.2 Constructor & Destructor Documentation	563
6.112.3 Member Function Documentation	563
6.112.4 Member Data Documentation	565
6.113Ipopt::MuOracle Class Reference	566
6.113.1 Detailed Description	567
6.113.2 Constructor & Destructor Documentation	567
6.113.3 Member Function Documentation	568
6.114Ipopt::MuUpdate Class Reference	568
6.114.1 Detailed Description	569
6.114.2 Constructor & Destructor Documentation	569
6.114.3 Member Function Documentation	569
6.115Ipopt::NLP Class Reference	570
6.115.1 Detailed Description	572
6.115.2 Constructor & Destructor Documentation	572
6.115.3 Member Function Documentation	572
6.116Ipopt::NLPBoundsRemover Class Reference	575
6.116.1 Detailed Description	577
6.116.2 Constructor & Destructor Documentation	577
6.116.3 Member Function Documentation	578
6.116.4 Member Data Documentation	580
6.117Ipopt::NLPScalingObject Class Reference	581
6.117.1 Detailed Description	583
6.117.2 Constructor & Destructor Documentation	584
6.117.3 Member Function Documentation	584
6.117.4 Member Data Documentation	588
6.118Ipopt::NoNLPScalingObject Class Reference	588
6.118.1 Detailed Description	589
6.118.2 Constructor & Destructor Documentation	589
6.118.3 Member Function Documentation	589
6.119Ipopt::Observer Class Reference	590
6.119.1 Detailed Description	591

6.119.2 Member Enumeration Documentation	591
6.119.3 Constructor & Destructor Documentation	591
6.119.4 Member Function Documentation	592
6.119.5 Friends And Related Function Documentation	592
6.119.6 Member Data Documentation	592
6.120Ipopt::OptimalityErrorConvergenceCheck Class Reference	593
6.120.1 Detailed Description	595
6.120.2 Constructor & Destructor Documentation	595
6.120.3 Member Function Documentation	595
6.120.4 Member Data Documentation	596
6.121Ipopt::OptionsList Class Reference	597
6.121.1 Detailed Description	599
6.121.2 Constructor & Destructor Documentation	600
6.121.3 Member Function Documentation	600
6.121.4 Member Data Documentation	602
6.122Ipopt::OptionsList::OptionValue Class Reference	602
6.122.1 Detailed Description	603
6.122.2 Constructor & Destructor Documentation	603
6.122.3 Member Function Documentation	604
6.122.4 Member Data Documentation	604
6.123Ipopt::OrigIpoptNLP Class Reference	605
6.123.1 Detailed Description	610
6.123.2 Constructor & Destructor Documentation	610
6.123.3 Member Function Documentation	610
6.123.4 Member Data Documentation	616
6.124Ipopt::OrigIterationOutput Class Reference	621
6.124.1 Detailed Description	622
6.124.2 Constructor & Destructor Documentation	622
6.124.3 Member Function Documentation	622
6.124.4 Member Data Documentation	623
6.125Ipopt::PardisoSolverInterface Class Reference	623
6.125.1 Detailed Description	626
6.125.2 Member Enumeration Documentation	626
6.125.3 Constructor & Destructor Documentation	627
6.125.4 Member Function Documentation	627
6.125.5 Member Data Documentation	628
6.126Ipopt::PDFullSpaceSolver Class Reference	631

6.126.1 Detailed Description	633
6.126.2 Constructor & Destructor Documentation	633
6.126.3 Member Function Documentation	633
6.126.4 Member Data Documentation	634
6.127Ipopt::PDPerturbationHandler Class Reference	635
6.127.1 Detailed Description	638
6.127.2 Member Enumeration Documentation	638
6.127.3 Constructor & Destructor Documentation	639
6.127.4 Member Function Documentation	639
6.127.5 Member Data Documentation	640
6.128Ipopt::PDSearchDirCalculator Class Reference	643
6.128.1 Detailed Description	644
6.128.2 Constructor & Destructor Documentation	644
6.128.3 Member Function Documentation	644
6.128.4 Member Data Documentation	645
6.129Ipopt::PDSysSolver Class Reference	645
6.129.1 Detailed Description	646
6.129.2 Constructor & Destructor Documentation	647
6.129.3 Member Function Documentation	647
6.130Ipopt::PenaltyLSAcceptor Class Reference	648
6.130.1 Detailed Description	650
6.130.2 Constructor & Destructor Documentation	651
6.130.3 Member Function Documentation	651
6.130.4 Member Data Documentation	653
6.131Ipopt::PiecewisePenalty Class Reference	655
6.131.1 Detailed Description	656
6.131.2 Constructor & Destructor Documentation	656
6.131.3 Member Function Documentation	657
6.131.4 Member Data Documentation	657
6.132Ipopt::PiecewisePenEntry Struct Reference	658
6.132.1 Detailed Description	658
6.132.2 Member Data Documentation	658
6.133Ipopt::PointPerturber Class Reference	659
6.133.1 Detailed Description	660
6.133.2 Constructor & Destructor Documentation	660
6.133.3 Member Function Documentation	660
6.133.4 Member Data Documentation	660

6.134Ipopt::AmplOptionsList::PrivatInfo Class Reference	660
6.134.1 Detailed Description	661
6.134.2 Constructor & Destructor Documentation	661
6.134.3 Member Function Documentation	661
6.134.4 Member Data Documentation	661
6.135Ipopt::ProbingMuOracle Class Reference	662
6.135.1 Detailed Description	663
6.135.2 Constructor & Destructor Documentation	663
6.135.3 Member Function Documentation	664
6.135.4 Member Data Documentation	664
6.136Ipopt::QualityFunctionMuOracle Class Reference	664
6.136.1 Detailed Description	667
6.136.2 Member Enumeration Documentation	668
6.136.3 Constructor & Destructor Documentation	668
6.136.4 Member Function Documentation	669
6.136.5 Member Data Documentation	670
6.137Ipopt::ReferencedObject Class Reference	673
6.137.1 Detailed Description	674
6.137.2 Constructor & Destructor Documentation	676
6.137.3 Member Function Documentation	676
6.137.4 Member Data Documentation	677
6.138Ipopt::Referencer Class Reference	677
6.138.1 Detailed Description	678
6.139Ipopt::RegisteredOption Class Reference	678
6.139.1 Detailed Description	681
6.139.2 Constructor & Destructor Documentation	681
6.139.3 Member Function Documentation	681
6.139.4 Member Data Documentation	685
6.140Ipopt::RegisteredOptions Class Reference	687
6.140.1 Detailed Description	689
6.140.2 Member Typedef Documentation	689
6.140.3 Constructor & Destructor Documentation	689
6.140.4 Member Function Documentation	690
6.140.5 Member Data Documentation	692
6.141Ipopt::RestoConvergenceCheck Class Reference	693
6.141.1 Detailed Description	694
6.141.2 Constructor & Destructor Documentation	695

6.141.3 Member Function Documentation	695
6.141.4 Member Data Documentation	696
6.142Ipopt::RestoFilterConvergenceCheck Class Reference	696
6.142.1 Detailed Description	698
6.142.2 Constructor & Destructor Documentation	698
6.142.3 Member Function Documentation	698
6.142.4 Member Data Documentation	699
6.143Ipopt::RestoIpoptNLP Class Reference	699
6.143.1 Detailed Description	703
6.143.2 Constructor & Destructor Documentation	704
6.143.3 Member Function Documentation	704
6.143.4 Member Data Documentation	709
6.144Ipopt::RestoIterateInitializer Class Reference	712
6.144.1 Detailed Description	714
6.144.2 Constructor & Destructor Documentation	714
6.144.3 Member Function Documentation	714
6.144.4 Member Data Documentation	715
6.145Ipopt::RestoIterationOutput Class Reference	715
6.145.1 Detailed Description	716
6.145.2 Constructor & Destructor Documentation	717
6.145.3 Member Function Documentation	717
6.145.4 Member Data Documentation	717
6.146Ipopt::RestoPenaltyConvergenceCheck Class Reference	718
6.146.1 Detailed Description	719
6.146.2 Constructor & Destructor Documentation	719
6.146.3 Member Function Documentation	720
6.146.4 Member Data Documentation	720
6.147Ipopt::RestoRestorationPhase Class Reference	721
6.147.1 Detailed Description	721
6.147.2 Constructor & Destructor Documentation	722
6.147.3 Member Function Documentation	722
6.148Ipopt::RestoRestorationPhase Class Reference	722
6.148.1 Detailed Description	723
6.148.2 Constructor & Destructor Documentation	724
6.148.3 Member Function Documentation	724
6.149Ipopt::ScaledMatrix Class Reference	724
6.149.1 Detailed Description	726

6.149.2 Constructor & Destructor Documentation	726
6.149.3 Member Function Documentation	727
6.149.4 Member Data Documentation	728
6.150Ipopt::ScaledMatrixSpace Class Reference	729
6.150.1 Detailed Description	730
6.150.2 Constructor & Destructor Documentation	730
6.150.3 Member Function Documentation	731
6.150.4 Member Data Documentation	731
6.151Ipopt::SearchDirectionCalculator Class Reference	732
6.151.1 Detailed Description	732
6.151.2 Constructor & Destructor Documentation	733
6.151.3 Member Function Documentation	733
6.152Ipopt::SlackBasedTSymScalingMethod Class Reference	733
6.152.1 Detailed Description	734
6.152.2 Constructor & Destructor Documentation	734
6.152.3 Member Function Documentation	735
6.153Ipopt::SmartPtr< T > Class Template Reference	735
6.153.1 Detailed Description	737
6.153.2 Constructor & Destructor Documentation	739
6.153.3 Member Function Documentation	739
6.153.4 Friends And Related Function Documentation	740
6.153.5 Member Data Documentation	741
6.154Ipopt::SolveStatistics Class Reference	742
6.154.1 Detailed Description	744
6.154.2 Constructor & Destructor Documentation	744
6.154.3 Member Function Documentation	744
6.154.4 Member Data Documentation	745
6.155Ipopt::SparseSymLinearSolverInterface Class Reference	747
6.155.1 Detailed Description	749
6.155.2 Member Enumeration Documentation	750
6.155.3 Constructor & Destructor Documentation	750
6.155.4 Member Function Documentation	751
6.156Ipopt::StandardScalingBase Class Reference	753
6.156.1 Detailed Description	755
6.156.2 Constructor & Destructor Documentation	755
6.156.3 Member Function Documentation	756
6.156.4 Member Data Documentation	759

6.157Ipopt::StdAugSystemSolver Class Reference	759
6.157.1 Detailed Description	762
6.157.2 Constructor & Destructor Documentation	762
6.157.3 Member Function Documentation	762
6.157.4 Member Data Documentation	764
6.158Ipopt::StdInterfaceTNLP Class Reference	766
6.158.1 Detailed Description	769
6.158.2 Constructor & Destructor Documentation	770
6.158.3 Member Function Documentation	770
6.158.4 Member Data Documentation	772
6.159Ipopt::StreamJournal Class Reference	775
6.159.1 Detailed Description	776
6.159.2 Constructor & Destructor Documentation	776
6.159.3 Member Function Documentation	777
6.159.4 Member Data Documentation	777
6.160Ipopt::RegisteredOption::string_entry Class Reference	777
6.160.1 Detailed Description	778
6.160.2 Constructor & Destructor Documentation	778
6.160.3 Member Data Documentation	778
6.161Ipopt::Subject Class Reference	778
6.161.1 Detailed Description	780
6.161.2 Constructor & Destructor Documentation	780
6.161.3 Member Function Documentation	780
6.161.4 Member Data Documentation	781
6.162Ipopt::SumMatrix Class Reference	781
6.162.1 Detailed Description	782
6.162.2 Constructor & Destructor Documentation	783
6.162.3 Member Function Documentation	783
6.162.4 Member Data Documentation	784
6.163Ipopt::SumMatrixSpace Class Reference	784
6.163.1 Detailed Description	786
6.163.2 Constructor & Destructor Documentation	786
6.163.3 Member Function Documentation	786
6.163.4 Member Data Documentation	787
6.164Ipopt::SumSymMatrix Class Reference	787
6.164.1 Detailed Description	788
6.164.2 Constructor & Destructor Documentation	789

6.164.3 Member Function Documentation	789
6.164.4 Member Data Documentation	790
6.165Ipopt::SumSymMatrixSpace Class Reference	790
6.165.1 Detailed Description	791
6.165.2 Constructor & Destructor Documentation	791
6.165.3 Member Function Documentation	792
6.165.4 Member Data Documentation	792
6.166Ipopt::SymLinearSolver Class Reference	792
6.166.1 Detailed Description	793
6.166.2 Constructor & Destructor Documentation	794
6.166.3 Member Function Documentation	794
6.167Ipopt::SymMatrix Class Reference	795
6.167.1 Detailed Description	796
6.167.2 Constructor & Destructor Documentation	796
6.167.3 Member Function Documentation	797
6.167.4 Member Data Documentation	797
6.168Ipopt::SymMatrixSpace Class Reference	797
6.168.1 Detailed Description	799
6.168.2 Constructor & Destructor Documentation	799
6.168.3 Member Function Documentation	799
6.169Ipopt::SymScaledMatrix Class Reference	800
6.169.1 Detailed Description	801
6.169.2 Constructor & Destructor Documentation	802
6.169.3 Member Function Documentation	802
6.169.4 Member Data Documentation	803
6.170Ipopt::SymScaledMatrixSpace Class Reference	803
6.170.1 Detailed Description	805
6.170.2 Constructor & Destructor Documentation	805
6.170.3 Member Function Documentation	805
6.170.4 Member Data Documentation	806
6.171Ipopt::SymTMatrix Class Reference	806
6.171.1 Detailed Description	808
6.171.2 Constructor & Destructor Documentation	809
6.171.3 Member Function Documentation	809
6.171.4 Member Data Documentation	810
6.172Ipopt::SymTMatrixSpace Class Reference	811
6.172.1 Detailed Description	812

6.172.2 Constructor & Destructor Documentation	812
6.172.3 Member Function Documentation	812
6.172.4 Friends And Related Function Documentation	813
6.172.5 Member Data Documentation	813
6.173Ipopt::TaggedObject Class Reference	813
6.173.1 Detailed Description	815
6.173.2 Member Typedef Documentation	816
6.173.3 Constructor & Destructor Documentation	816
6.173.4 Member Function Documentation	816
6.173.5 Member Data Documentation	816
6.174Ipopt::TDependencyDetector Class Reference	817
6.174.1 Detailed Description	818
6.174.2 Constructor & Destructor Documentation	818
6.174.3 Member Function Documentation	818
6.175Ipopt::TimedTask Class Reference	819
6.175.1 Detailed Description	820
6.175.2 Constructor & Destructor Documentation	820
6.175.3 Member Function Documentation	820
6.175.4 Member Data Documentation	821
6.176Ipopt::TimingStatistics Class Reference	822
6.176.1 Detailed Description	824
6.176.2 Constructor & Destructor Documentation	824
6.176.3 Member Function Documentation	824
6.176.4 Member Data Documentation	827
6.177Ipopt::TNLP Class Reference	829
6.177.1 Detailed Description	831
6.177.2 Member Typedef Documentation	831
6.177.3 Member Enumeration Documentation	832
6.177.4 Constructor & Destructor Documentation	832
6.177.5 Member Function Documentation	832
6.178Ipopt::TNLPAdapter Class Reference	836
6.178.1 Detailed Description	841
6.178.2 Member Enumeration Documentation	841
6.178.3 Constructor & Destructor Documentation	842
6.178.4 Member Function Documentation	842
6.178.5 Member Data Documentation	845
6.179Ipopt::TNLPReducer Class Reference	852

6.179.1 Detailed Description	854
6.179.2 Constructor & Destructor Documentation	854
6.179.3 Member Function Documentation	855
6.179.4 Member Data Documentation	857
6.180Ipopt::TransposeMatrix Class Reference	859
6.180.1 Detailed Description	860
6.180.2 Constructor & Destructor Documentation	860
6.180.3 Member Function Documentation	860
6.180.4 Member Data Documentation	862
6.181Ipopt::TransposeMatrixSpace Class Reference	862
6.181.1 Detailed Description	863
6.181.2 Constructor & Destructor Documentation	863
6.181.3 Member Function Documentation	863
6.181.4 Member Data Documentation	864
6.182Ipopt::TripletToCSRConverter::TripletEntry Class Reference	864
6.182.1 Detailed Description	865
6.182.2 Constructor & Destructor Documentation	865
6.182.3 Member Function Documentation	865
6.182.4 Member Data Documentation	866
6.183Ipopt::TripletHelper Class Reference	866
6.183.1 Detailed Description	868
6.183.2 Member Function Documentation	868
6.184Ipopt::TripletToCSRConverter Class Reference	870
6.184.1 Detailed Description	872
6.184.2 Member Enumeration Documentation	872
6.184.3 Constructor & Destructor Documentation	872
6.184.4 Member Function Documentation	873
6.184.5 Member Data Documentation	873
6.185Ipopt::TSymDependencyDetector Class Reference	875
6.185.1 Detailed Description	876
6.185.2 Constructor & Destructor Documentation	876
6.185.3 Member Function Documentation	876
6.185.4 Member Data Documentation	877
6.186Ipopt::TSymLinearSolver Class Reference	877
6.186.1 Detailed Description	880
6.186.2 Constructor & Destructor Documentation	880
6.186.3 Member Function Documentation	880

6.186.4 Member Data Documentation	881
6.187Ipopt::TSymScalingMethod Class Reference	883
6.187.1 Detailed Description	884
6.187.2 Constructor & Destructor Documentation	884
6.187.3 Member Function Documentation	884
6.188Ipopt::UserScaling Class Reference	885
6.188.1 Detailed Description	886
6.188.2 Constructor & Destructor Documentation	886
6.188.3 Member Function Documentation	886
6.188.4 Member Data Documentation	887
6.189Ipopt::Vector Class Reference	887
6.189.1 Detailed Description	891
6.189.2 Constructor & Destructor Documentation	891
6.189.3 Member Function Documentation	892
6.189.4 Member Data Documentation	898
6.190Ipopt::VectorSpace Class Reference	899
6.190.1 Detailed Description	900
6.190.2 Constructor & Destructor Documentation	900
6.190.3 Member Function Documentation	901
6.190.4 Member Data Documentation	901
6.191Ipopt::WarmStartIterateInitializer Class Reference	901
6.191.1 Detailed Description	903
6.191.2 Constructor & Destructor Documentation	903
6.191.3 Member Function Documentation	903
6.191.4 Member Data Documentation	904
6.192Ipopt::WsmpSolverInterface Class Reference	905
6.192.1 Detailed Description	907
6.192.2 Constructor & Destructor Documentation	908
6.192.3 Member Function Documentation	908
6.192.4 Member Data Documentation	909
6.193Ipopt::ZeroMatrix Class Reference	912
6.193.1 Detailed Description	913
6.193.2 Constructor & Destructor Documentation	913
6.193.3 Member Function Documentation	914
6.194Ipopt::ZeroMatrixSpace Class Reference	914
6.194.1 Detailed Description	915
6.194.2 Constructor & Destructor Documentation	915

6.194.3 Member Function Documentation	916
6.195 Ipopt::ZeroSymMatrix Class Reference	916
6.195.1 Detailed Description	918
6.195.2 Constructor & Destructor Documentation	918
6.195.3 Member Function Documentation	918
6.196 Ipopt::ZeroSymMatrixSpace Class Reference	919
6.196.1 Detailed Description	920
6.196.2 Constructor & Destructor Documentation	920
6.196.3 Member Function Documentation	920
7 File Documentation	921
7.1 Algorithm/Inexact/IpInexactAlgBuilder.hpp File Reference	921
7.2 Algorithm/Inexact/IpInexactCq.hpp File Reference	921
7.3 Algorithm/Inexact/IpInexactData.hpp File Reference	922
7.4 Algorithm/Inexact/IpInexactDoglegNormal.hpp File Reference	922
7.5 Algorithm/Inexact/IpInexactLSAcceptor.hpp File Reference	922
7.6 Algorithm/Inexact/IpInexactNewtonNormal.hpp File Reference	923
7.7 Algorithm/Inexact/IpInexactNormalStepCalc.hpp File Reference	923
7.8 Algorithm/Inexact/IpInexactNormalTerminationTester.hpp File Reference	923
7.9 Algorithm/Inexact/IpInexactPDSolver.hpp File Reference	924
7.10 Algorithm/Inexact/IpInexactPDTerminationTester.hpp File Reference	924
7.11 Algorithm/Inexact/IpInexactRegOp.hpp File Reference	924
7.12 Algorithm/Inexact/IpInexactSearchDirCalc.hpp File Reference	925
7.13 Algorithm/Inexact/IpInexactTSymScalingMethod.hpp File Reference	925
7.14 Algorithm/Inexact/IpIterativePardisoSolverInterface.hpp File Reference	925
7.15 Algorithm/Inexact/IpIterativeSolverTerminationTester.hpp File Reference	926
7.16 Algorithm/IpAdaptiveMuUpdate.hpp File Reference	926
7.17 Algorithm/IpAlgBuilder.hpp File Reference	926
7.18 Algorithm/IpAlgorithmRegOp.hpp File Reference	927
7.19 Algorithm/IpAlgStrategy.hpp File Reference	927
7.20 Algorithm/IpAugRestoSystemSolver.hpp File Reference	927
7.21 Algorithm/IpAugSystemSolver.hpp File Reference	928
7.22 Algorithm/IpBacktrackingLineSearch.hpp File Reference	928
7.23 Algorithm/IpBacktrackingLSAcceptor.hpp File Reference	929
7.24 Algorithm/IpConvCheck.hpp File Reference	929
7.25 Algorithm/IpDefaultIterateInitializer.hpp File Reference	929
7.26 Algorithm/IpEqMultCalculator.hpp File Reference	930

7.27	Algorithm/IpEquilibrationScaling.hpp File Reference	930
7.28	Algorithm/IpExactHessianUpdater.hpp File Reference	930
7.29	Algorithm/IpFilter.hpp File Reference	931
7.30	Algorithm/IpFilterLSAcceptor.hpp File Reference	931
7.31	Algorithm/IpGenAugSystemSolver.hpp File Reference	931
7.32	Algorithm/IpGradientScaling.hpp File Reference	932
7.33	Algorithm/IpHessianUpdater.hpp File Reference	932
7.34	Algorithm/IpIpoptAlg.hpp File Reference	932
7.35	Algorithm/IpIpoptCalculatedQuantities.hpp File Reference	933
7.36	Algorithm/IpIpoptData.hpp File Reference	933
7.37	Algorithm/IpIpoptNLP.hpp File Reference	934
7.38	Algorithm/IpIterateInitializer.hpp File Reference	934
7.39	Algorithm/IpIteratesVector.hpp File Reference	935
7.40	Algorithm/IpIterationOutput.hpp File Reference	935
7.41	Algorithm/IpLeastSquareMults.hpp File Reference	935
7.42	Algorithm/IpLimMemQuasiNewtonUpdater.hpp File Reference	936
7.43	Algorithm/IpLineSearch.hpp File Reference	936
7.44	Algorithm/IpLoqoMuOracle.hpp File Reference	936
7.45	Algorithm/IpLowRankAugSystemSolver.hpp File Reference	937
7.46	Algorithm/IpLowRankSSAugSystemSolver.hpp File Reference	937
7.47	Algorithm/IpMonotoneMuUpdate.hpp File Reference	937
7.48	Algorithm/IpMuOracle.hpp File Reference	938
7.49	Algorithm/IpMuUpdate.hpp File Reference	938
7.50	Algorithm/IpNLPBoundsRemover.hpp File Reference	938
7.51	Algorithm/IpNLPScaling.hpp File Reference	939
7.52	Algorithm/IpOptErrorConvCheck.hpp File Reference	939
7.53	Algorithm/IpOrigIpoptNLP.hpp File Reference	939
7.54	Algorithm/IpOrigIterationOutput.hpp File Reference	940
7.55	Algorithm/IpPDFullSpaceSolver.hpp File Reference	940
7.56	Algorithm/IpPDPerturbationHandler.hpp File Reference	940
7.57	Algorithm/IpPDSearchDirCalc.hpp File Reference	941
7.58	Algorithm/IpPDSysSystemSolver.hpp File Reference	941
7.59	Algorithm/IpPenaltyLSAcceptor.hpp File Reference	941
7.60	Algorithm/IpProbingMuOracle.hpp File Reference	942
7.61	Algorithm/IpQualityFunctionMuOracle.hpp File Reference	942
7.62	Algorithm/IpRestoConvCheck.hpp File Reference	942
7.63	Algorithm/IpRestoFilterConvCheck.hpp File Reference	943

7.64	Algorithm/IpRestoIpoptNLP.hpp File Reference	943
7.65	Algorithm/IpRestoIterateInitializer.hpp File Reference	943
7.66	Algorithm/IpRestoIterationOutput.hpp File Reference	944
7.67	Algorithm/IpRestoMinC_1Nrm.hpp File Reference	944
7.68	Algorithm/IpRestoPenaltyConvCheck.hpp File Reference	944
7.69	Algorithm/IpRestoPhase.hpp File Reference	945
7.70	Algorithm/IpRestoRestoPhase.hpp File Reference	945
7.71	Algorithm/IpSearchDirCalculator.hpp File Reference	946
7.72	Algorithm/IpStdAugSystemSolver.hpp File Reference	946
7.73	Algorithm/IpTimingStatistics.hpp File Reference	946
7.74	Algorithm/IpUserScaling.hpp File Reference	947
7.75	Algorithm/IpWarmStartIterateInitializer.hpp File Reference	947
7.76	Algorithm/LinearSolvers/hsl_ma77d.h File Reference	947
7.76.1	Macro Definition Documentation	949
7.76.2	Typedef Documentation	950
7.76.3	Function Documentation	950
7.77	Algorithm/LinearSolvers/hsl_ma86d.h File Reference	951
7.77.1	Macro Definition Documentation	952
7.77.2	Typedef Documentation	953
7.77.3	Function Documentation	953
7.78	Algorithm/LinearSolvers/hsl_ma97d.h File Reference	953
7.78.1	Macro Definition Documentation	954
7.78.2	Typedef Documentation	956
7.78.3	Function Documentation	956
7.79	Algorithm/LinearSolvers/hsl_mc68i.h File Reference	957
7.79.1	Macro Definition Documentation	957
7.79.2	Function Documentation	958
7.80	Algorithm/LinearSolvers/IpGenKKTsSolverInterface.hpp File Reference	958
7.81	Algorithm/LinearSolvers/IpIterativeWsmpSolverInterface.hpp File Reference	958
7.82	Algorithm/LinearSolvers/IpLinearSolversRegOp.hpp File Reference	958
7.83	Algorithm/LinearSolvers/IpMa27TSolverInterface.hpp File Reference	959
7.84	Algorithm/LinearSolvers/IpMa28TDependencyDetector.hpp File Reference	959
7.85	Algorithm/LinearSolvers/IpMa57TSolverInterface.hpp File Reference	959
7.85.1	Typedef Documentation	960
7.86	Algorithm/LinearSolvers/IpMa77SolverInterface.hpp File Reference	960
7.87	Algorithm/LinearSolvers/IpMa86SolverInterface.hpp File Reference	960
7.88	Algorithm/LinearSolvers/IpMa97SolverInterface.hpp File Reference	960

7.89 Algorithm/LinearSolvers/lpMc19TSymScalingMethod.hpp File Reference	961
7.90 Algorithm/LinearSolvers/lpMumpsSolverInterface.hpp File Reference	961
7.91 Algorithm/LinearSolvers/lpPardisoSolverInterface.hpp File Reference	961
7.92 Algorithm/LinearSolvers/lpSlackBasedTSymScalingMethod.hpp File Reference	962
7.93 Algorithm/LinearSolvers/lpSparseSymLinearSolverInterface.hpp File Reference	962
7.94 Algorithm/LinearSolvers/lpSymLinearSolver.hpp File Reference	962
7.95 Algorithm/LinearSolvers/lpTDependencyDetector.hpp File Reference	963
7.96 Algorithm/LinearSolvers/lpTripletToCSRConverter.hpp File Reference	963
7.97 Algorithm/LinearSolvers/lpTSymDependencyDetector.hpp File Reference	964
7.98 Algorithm/LinearSolvers/lpTSymLinearSolver.hpp File Reference	964
7.99 Algorithm/LinearSolvers/lpTSymScalingMethod.hpp File Reference	964
7.100 Algorithm/LinearSolvers/lpWsmpSolverInterface.hpp File Reference	965
7.101 Apps/AmplSolver/AmplTNLP.hpp File Reference	965
7.102 Common/config_default.h File Reference	966
7.102.1 Macro Definition Documentation	966
7.103 Common/config_ipopt_default.h File Reference	966
7.103.1 Macro Definition Documentation	967
7.104 Common/lpCachedResults.hpp File Reference	967
7.105 Common/lpDebug.hpp File Reference	967
7.105.1 Macro Definition Documentation	968
7.106 Common/lpException.hpp File Reference	969
7.106.1 Macro Definition Documentation	969
7.107 Common/lpJournalist.hpp File Reference	970
7.108 Common/lpObserver.hpp File Reference	971
7.109 Common/lpoptConfig.h File Reference	971
7.110 Common/lpOptionsList.hpp File Reference	971
7.111 Common/lpReferenced.hpp File Reference	972
7.112 Common/lpRegOptions.hpp File Reference	972
7.113 Common/lpSmartPtr.hpp File Reference	973
7.113.1 Macro Definition Documentation	974
7.114 Common/lpTaggedObject.hpp File Reference	974
7.115 Common/lpTimedTask.hpp File Reference	975
7.116 Common/lpTypes.hpp File Reference	975
7.116.1 Typedef Documentation	975
7.117 Common/lpUtils.hpp File Reference	976
7.118 contrib/CGPenalty/lpCGPenaltyCq.hpp File Reference	976
7.119 contrib/CGPenalty/lpCGPenaltyData.hpp File Reference	977

7.120contrib/CGPenalty/lpCGPenaltyLSAcceptor.hpp File Reference	977
7.121contrib/CGPenalty/lpCGPenaltyRegOp.hpp File Reference	977
7.122contrib/CGPenalty/lpCGPerturbationHandler.hpp File Reference	978
7.123contrib/CGPenalty/lpCGSearchDirCalc.hpp File Reference	978
7.124contrib/CGPenalty/lpPiecewisePenalty.hpp File Reference	979
7.125contrib/LinearSolverLoader/HSLLoader.h File Reference	979
7.125.1 Macro Definition Documentation	983
7.125.2 Typedef Documentation	985
7.125.3 Function Documentation	989
7.126contrib/LinearSolverLoader/LibraryHandler.h File Reference	992
7.126.1 Typedef Documentation	992
7.126.2 Function Documentation	993
7.127contrib/LinearSolverLoader/PardisoLoader.h File Reference	993
7.127.1 Function Documentation	993
7.128Interfaces/lpAlgTypes.hpp File Reference	994
7.129Interfaces/lpInterfacesRegOp.hpp File Reference	995
7.130Interfaces/lpIpoptApplication.hpp File Reference	995
7.130.1 Macro Definition Documentation	996
7.130.2 Function Documentation	996
7.131Interfaces/lpNLP.hpp File Reference	996
7.132Interfaces/lpReturnCodes.h File Reference	996
7.133Interfaces/lpReturnCodes.hpp File Reference	996
7.134Interfaces/lpReturnCodes_inc.h File Reference	997
7.134.1 Enumeration Type Documentation	997
7.135Interfaces/lpSolveStatistics.hpp File Reference	998
7.136Interfaces/lpStdCInterface.h File Reference	998
7.136.1 Macro Definition Documentation	1001
7.136.2 Typedef Documentation	1001
7.136.3 Function Documentation	1003
7.136.4 Variable Documentation	1004
7.137Interfaces/lpStdInterfaceTNLP.hpp File Reference	1007
7.138Interfaces/lpTNLP.hpp File Reference	1007
7.139Interfaces/lpTNLPAdapter.hpp File Reference	1008
7.140Interfaces/lpTNLPReducer.hpp File Reference	1008
7.141LinAlg/lpBlas.hpp File Reference	1008
7.142LinAlg/lpCompoundMatrix.hpp File Reference	1009
7.143LinAlg/lpCompoundSymMatrix.hpp File Reference	1009

7.144LinAlg/lpCompoundVector.hpp File Reference	1010
7.145LinAlg/lpDenseGenMatrix.hpp File Reference	1010
7.146LinAlg/lpDenseSymMatrix.hpp File Reference	1011
7.147LinAlg/lpDenseVector.hpp File Reference	1011
7.148LinAlg/lpDiagMatrix.hpp File Reference	1012
7.149LinAlg/lpExpandedMultiVectorMatrix.hpp File Reference	1012
7.150LinAlg/lpExpansionMatrix.hpp File Reference	1012
7.151LinAlg/lpIdentityMatrix.hpp File Reference	1013
7.152LinAlg/lpLapack.hpp File Reference	1013
7.153LinAlg/lpLowRankUpdateSymMatrix.hpp File Reference	1014
7.154LinAlg/lpMatrix.hpp File Reference	1014
7.154.1 Macro Definition Documentation	1015
7.155LinAlg/lpMultiVectorMatrix.hpp File Reference	1015
7.156LinAlg/lpScaledMatrix.hpp File Reference	1015
7.157LinAlg/lpSumMatrix.hpp File Reference	1015
7.158LinAlg/lpSumSymMatrix.hpp File Reference	1016
7.159LinAlg/lpSymMatrix.hpp File Reference	1016
7.160LinAlg/lpSymScaledMatrix.hpp File Reference	1017
7.161LinAlg/lpTransposeMatrix.hpp File Reference	1017
7.162LinAlg/lpVector.hpp File Reference	1017
7.162.1 Macro Definition Documentation	1018
7.163LinAlg/lpZeroMatrix.hpp File Reference	1018
7.164LinAlg/lpZeroSymMatrix.hpp File Reference	1018
7.165LinAlg/TMatrices/lpGenTMatrix.hpp File Reference	1019
7.166LinAlg/TMatrices/lpSymTMatrix.hpp File Reference	1019
7.167LinAlg/TMatrices/lpTripletHelper.hpp File Reference	1020

Index

1021

1 Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

lpopt

32

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

std::allocator< T >	
std::array< T >	
std::auto_ptr< T >	
std::basic_string< Char >	
std::string	
std::wstring	
std::basic_string< char >	
std::basic_string< wchar_t >	
std::bitset< Bits >	
Ipopt::CachedResults< T >	112
Ipopt::CachedResults< Ipopt::SmartPtr< const Ipopt::Matrix > >	112
Ipopt::CachedResults< Ipopt::SmartPtr< const Ipopt::SymMatrix > >	112
Ipopt::CachedResults< Ipopt::SmartPtr< const Ipopt::Vector > >	112
Ipopt::CachedResults< Ipopt::SmartPtr< Ipopt::Vector > >	112
Ipopt::CachedResults< Number >	112
Ipopt::CachedResults< void * >	112
std::complex	
std::list< T >::const_iterator	
std::forward_list< T >::const_iterator	
std::map< K, T >::const_iterator	
std::unordered_map< K, T >::const_iterator	
std::multimap< K, T >::const_iterator	
std::basic_string< Char >::const_iterator	
std::unordered_multimap< K, T >::const_iterator	
std::set< K >::const_iterator	
std::string::const_iterator	
std::unordered_set< K >::const_iterator	
std::wstring::const_iterator	
std::multiset< K >::const_iterator	
std::unordered_multiset< K >::const_iterator	
std::vector< T >::const_iterator	
std::deque< T >::const_iterator	
std::list< T >::const_reverse_iterator	
std::forward_list< T >::const_reverse_iterator	
std::map< K, T >::const_reverse_iterator	
std::unordered_map< K, T >::const_reverse_iterator	
std::multimap< K, T >::const_reverse_iterator	
std::basic_string< Char >::const_reverse_iterator	
std::unordered_multimap< K, T >::const_reverse_iterator	
std::set< K >::const_reverse_iterator	
std::string::const_reverse_iterator	
std::unordered_set< K >::const_reverse_iterator	

```

std::multiset< K >::const_reverse_iterator
std::wstring::const_reverse_iterator
std::unordered_multiset< K >::const_reverse_iterator
std::vector< T >::const_reverse_iterator
std::deque< T >::const_reverse_iterator
std::deque< T >
std::error_category
std::error_code
std::error_condition
std::exception
    std::bad_alloc
    std::bad_cast
    std::bad_exception
    std::bad_typeid
    std::ios_base::failure
    std::logic_error
        std::domain_error
        std::invalid_argument
        std::length_error
        std::out_of_range
    std::runtime_error
        std::overflow_error
        std::range_error
        std::underflow_error

```

Ipopt::Filter**242****Ipopt::FilterEntry****244**

```

std::forward_list< T >
std::ios_base
    basic_ios< char >
    basic_ios< wchar_t >
    std::basic_ios
        basic_istream< char >
        basic_istream< wchar_t >
        basic_ostream< char >
        basic_ostream< wchar_t >
        std::basic_istream
            basic_ifstream< char >
            basic_ifstream< wchar_t >
            basic_iostream< char >
            basic_iostream< wchar_t >
            basic_istreamstream< char >
            basic_istreamstream< wchar_t >
            std::basic_ifstream
                std::ifstream
                std::wifstream
            std::basic_iostream
                basic_fstream< char >
                basic_fstream< wchar_t >
                basic_stringstream< char >
                basic_stringstream< wchar_t >
                std::basic_fstream
                std::fstream

```

```

        std::wfstream
    std::basic_stringstream
        std::stringstream
        std::wstringstream
    std::basic_istream
        std::istream
        std::wistream
    std::basic_ostream
        basic_iostream< char >
        basic_iostream< wchar_t >
        basic_ofstream< char >
        basic_ofstream< wchar_t >
        basic_ostringstream< char >
        basic_ostringstream< wchar_t >
    std::basic_iostream
    std::basic_ofstream
        std::ofstream
        std::wofstream
    std::basic_ostringstream
        std::ostringstream
        std::wostringstream
    std::ostream
    std::wostream
    std::ios
    std::wios

```

Ipopt::IpoptException

389

```

std::list< T >::iterator
std::forward_list< T >::iterator
std::map< K, T >::iterator
std::unordered_map< K, T >::iterator
std::multimap< K, T >::iterator
std::basic_string< Char >::iterator
std::unordered_multimap< K, T >::iterator
std::set< K >::iterator
std::string::iterator
std::unordered_set< K >::iterator
std::wstring::iterator
std::multiset< K >::iterator
std::unordered_multiset< K >::iterator
std::vector< T >::iterator
std::deque< T >::iterator
std::list< T >
std::list< Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::Matrix > > * >
std::list< Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::SymMatrix > > * >
std::list< Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::Vector > > * >
std::list< Ipopt::DependentResult< Ipopt::SmartPtr< Ipopt::Vector > > * >
std::list< Ipopt::DependentResult< Number > * >
std::list< Ipopt::DependentResult< T > * >
std::list< Ipopt::DependentResult< void * > * >
std::list< Ipopt::FilterEntry * >
std::list< Number >

```

ma77_control_d	500
ma77_info_d	503
ma86_control_d	512
ma86_info_d	514
ma97_control_d	521
ma97_info	523
std::map< K, T >	
std::map< std::string, Ipopt::OptionsList::OptionValue >	
std::map< std::string, Ipopt::SmartPtr< const Ipopt::AmplOptionsList::AmplOption > >	
std::map< std::string, Ipopt::SmartPtr< Ipopt::RegisteredOption > >	
std::map< std::string, std::vector< Index > >	
std::map< std::string, std::vector< Number > >	
std::map< std::string, std::vector< std::string > >	
mc68_control	543
mc68_info	545
std::multimap< K, T >	
std::multiset< K >	
Ipopt::Observer	590
Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::Matrix > >	213
Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::SymMatrix > >	213
Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::Vector > >	213
Ipopt::DependentResult< Ipopt::SmartPtr< Ipopt::Vector > >	213
Ipopt::DependentResult< Number >	213
Ipopt::DependentResult< void * >	213
Ipopt::DependentResult< T >	213
Ipopt::OptionsList::OptionValue	602
Ipopt::PiecewisePenalty	655
Ipopt::PiecewisePenEntry	658
std::priority_queue< T >	
Ipopt::AmplOptionsList::PrivatInfo	660
std::queue< T >	
Ipopt::ReferencedObject	673
Ipopt::AlgorithmBuilder	63
Ipopt::InexactAlgorithmBuilder	281
Ipopt::AlgorithmStrategyObject	65

Ipopt::AugSystemSolver	92
Ipopt::AugRestoSystemSolver	87
Ipopt::GenAugSystemSolver	256
Ipopt::LowRankAugSystemSolver	463
Ipopt::LowRankSSAugSystemSolver	470
Ipopt::StdAugSystemSolver	759
Ipopt::BacktrackingLSAcceptor	107
Ipopt::CGPenaltyLSAcceptor	128
Ipopt::FilterLSAcceptor	247
Ipopt::InexactLSAcceptor	297
Ipopt::PenaltyLSAcceptor	648
Ipopt::ConvergenceCheck	180
Ipopt::OptimalityErrorConvergenceCheck	593
Ipopt::RestoConvergenceCheck	693
Ipopt::RestoFilterConvergenceCheck	696
Ipopt::RestoPenaltyConvergenceCheck	718
Ipopt::EqMultiplierCalculator	221
Ipopt::LeastSquareMultipliers	444
Ipopt::GenKKTsSolverInterface	262
Ipopt::HessianUpdater	275
Ipopt::ExactHessianUpdater	226
Ipopt::LimMemQuasiNewtonUpdater	446
Ipopt::InexactNewtonNormalStep	305
Ipopt::InexactNormalStepCalculator	308
Ipopt::InexactDoglegNormalStep	294
Ipopt::InexactPDSolver	313
Ipopt::IpoptAlgorithm	332
Ipopt::IterateInitializer	399
Ipopt::DefaultIterateInitializer	182
Ipopt::RestoIterateInitializer	712

Ipopt::WarmStartIterateInitializer	901
Ipopt::IterationOutput	415
Ipopt::OrigIterationOutput	621
Ipopt::RestoIterationOutput	715
Ipopt::IterativeSolverTerminationTester	427
Ipopt::InexactNormalTerminationTester	310
Ipopt::InexactPDTerminationTester	317
Ipopt::LineSearch	459
Ipopt::BacktrackingLineSearch	96
Ipopt::MuOracle	566
Ipopt::LoqoMuOracle	461
Ipopt::ProbingMuOracle	662
Ipopt::QualityFunctionMuOracle	664
Ipopt::MuUpdate	568
Ipopt::AdaptiveMuUpdate	55
Ipopt::MonotoneMuUpdate	550
Ipopt::PDPerturbationHandler	635
Ipopt::CGPerturbationHandler	139
Ipopt::PDSysSystemSolver	645
Ipopt::PDFullSpaceSolver	631
Ipopt::RestorationPhase	721
Ipopt::MinC_1NrmRestorationPhase	546
Ipopt::RestoRestorationPhase	722
Ipopt::SearchDirectionCalculator	732
Ipopt::CGSearchDirCalculator	146
Ipopt::InexactSearchDirCalculator	323
Ipopt::PDSearchDirCalculator	643
Ipopt::SparseSymLinearSolverInterface	747
Ipopt::IterativePardisoSolverInterface	417
Ipopt::IterativeWsmvSolverInterface	430

Ipopt::Ma27TSolverInterface	484
Ipopt::Ma57TSolverInterface	494
Ipopt::Ma77TSolverInterface	507
Ipopt::Ma86TSolverInterface	515
Ipopt::Ma97TSolverInterface	525
Ipopt::MumpsSolverInterface	560
Ipopt::PardisoSolverInterface	623
Ipopt::WsmvSolverInterface	905
Ipopt::SymLinearSolver	792
Ipopt::TSymLinearSolver	877
Ipopt::TDependencyDetector	817
Ipopt::Ma28TDependencyDetector	491
Ipopt::TSymDependencyDetector	875
Ipopt::TSymScalingMethod	883
Ipopt::InexactTSymScalingMethod	327
Ipopt::Mc19TSymScalingMethod	542
Ipopt::SlackBasedTSymScalingMethod	733
Ipopt::AmplOptionsList	72
Ipopt::AmplOptionsList::AmplOption	70
Ipopt::AmplSuffixHandler	74
Ipopt::IpoptAdditionalCq	329
Ipopt::CGPenaltyCq	117
Ipopt::InexactCq	284
Ipopt::IpoptAdditionalData	330
Ipopt::CGPenaltyData	122
Ipopt::InexactData	289
Ipopt::IpoptApplication	338
Ipopt::IpoptCalculatedQuantities	345
Ipopt::IpoptData	374
Ipopt::IpoptNLP	391

Ipopt::OrigIpoptNLP	605
Ipopt::RestolIpoptNLP	699
Ipopt::Journal	436
Ipopt::FileJournal	240
Ipopt::StreamJournal	775
Ipopt::Journalist	440
Ipopt::MatrixSpace	539
Ipopt::CompoundMatrixSpace	156
Ipopt::DenseGenMatrixSpace	194
Ipopt::ExpandedMultiVectorMatrixSpace	231
Ipopt::ExpansionMatrixSpace	237
Ipopt::GenTMatrixSpace	269
Ipopt::MultiVectorMatrixSpace	558
Ipopt::ScaledMatrixSpace	729
Ipopt::SumMatrixSpace	784
Ipopt::SymMatrixSpace	797
Ipopt::CompoundSymMatrixSpace	165
Ipopt::DenseSymMatrixSpace	199
Ipopt::DiagMatrixSpace	219
Ipopt::IdentityMatrixSpace	280
Ipopt::LowRankUpdateSymMatrixSpace	481
Ipopt::SumSymMatrixSpace	790
Ipopt::SymScaledMatrixSpace	803
Ipopt::SymTMatrixSpace	811
Ipopt::ZeroSymMatrixSpace	919
Ipopt::TransposeMatrixSpace	862
Ipopt::ZeroMatrixSpace	914
Ipopt::NLP	570
Ipopt::NLPBoundsRemover	575
Ipopt::TNLPAdapter	836

Ipopt::NLPScalingObject	581
Ipopt::StandardScalingBase	753
Ipopt::EquilibrationScaling	223
Ipopt::GradientScaling	272
Ipopt::NoNLPScalingObject	588
Ipopt::UserScaling	885
Ipopt::OptionsList	597
Ipopt::PointPerturber	659
Ipopt::RegisteredOption	678
Ipopt::RegisteredOptions	687
Ipopt::SolveStatistics	742
Ipopt::TaggedObject	813
Ipopt::Matrix	532
Ipopt::CompoundMatrix	151
Ipopt::DenseGenMatrix	188
Ipopt::ExpandedMultiVectorMatrix	227
Ipopt::ExpansionMatrix	233
Ipopt::GenTMatrix	264
Ipopt::MultiVectorMatrix	553
Ipopt::ScaledMatrix	724
Ipopt::SumMatrix	781
Ipopt::SymMatrix	795
Ipopt::CompoundSymMatrix	160
Ipopt::DenseSymMatrix	195
Ipopt::DiagMatrix	216
Ipopt::IdentityMatrix	276
Ipopt::LowRankUpdateSymMatrix	476
Ipopt::SumSymMatrix	787
Ipopt::SymScaledMatrix	800
Ipopt::SymTMatrix	806

Ipopt::ZeroSymMatrix	916
Ipopt::TransposeMatrix	859
Ipopt::ZeroMatrix	912
Ipopt::Vector	887
Ipopt::CompoundVector	168
Ipopt::IteratesVector	400
Ipopt::DenseVector	200
Ipopt::TimingStatistics	822
Ipopt::TNLP	829
Ipopt::AmplTNLP	77
Ipopt::StdInterfaceTNLP	766
Ipopt::TNLPReducer	852
Ipopt::TripletToCSRConverter	870
Ipopt::VectorSpace	899
Ipopt::CompoundVectorSpace	176
Ipopt::IteratesVectorSpace	412
Ipopt::DenseVectorSpace	209
Ipopt::Referencer	677
Ipopt::SmartPtr< T >	735
Ipopt::SmartPtr< const Ipopt::AmplOptionsList::AmplOption >	735
Ipopt::SmartPtr< const Ipopt::CompoundVectorSpace >	735
Ipopt::SmartPtr< const Ipopt::ExpansionMatrix >	735
Ipopt::SmartPtr< const Ipopt::IteratesVector >	735
Ipopt::SmartPtr< const Ipopt::Journalist >	735
Ipopt::SmartPtr< const Ipopt::LowRankUpdateSymMatrixSpace >	735
Ipopt::SmartPtr< const Ipopt::Matrix >	735
Ipopt::SmartPtr< const Ipopt::MatrixSpace >	735
Ipopt::SmartPtr< const Ipopt::MultiVectorMatrix >	735
Ipopt::SmartPtr< const Ipopt::NLP >	735
Ipopt::SmartPtr< const Ipopt::ScaledMatrixSpace >	735

<code>Ipopt::SmartPtr< const Ipopt::SymMatrix ></code>	735
<code>Ipopt::SmartPtr< const Ipopt::SymMatrixSpace ></code>	735
<code>Ipopt::SmartPtr< const Ipopt::SymScaledMatrixSpace ></code>	735
<code>Ipopt::SmartPtr< const Ipopt::Vector ></code>	735
<code>Ipopt::SmartPtr< const Ipopt::VectorSpace ></code>	735
<code>Ipopt::SmartPtr< Ipopt::AmplSuffixHandler ></code>	735
<code>Ipopt::SmartPtr< Ipopt::AugSystemSolver ></code>	735
<code>Ipopt::SmartPtr< Ipopt::BacktrackingLSAcceptor ></code>	735
<code>Ipopt::SmartPtr< Ipopt::CompoundMatrix ></code>	735
<code>Ipopt::SmartPtr< Ipopt::CompoundMatrixSpace ></code>	735
<code>Ipopt::SmartPtr< Ipopt::CompoundSymMatrix ></code>	735
<code>Ipopt::SmartPtr< Ipopt::CompoundSymMatrixSpace ></code>	735
<code>Ipopt::SmartPtr< Ipopt::CompoundVector ></code>	735
<code>Ipopt::SmartPtr< Ipopt::CompoundVectorSpace ></code>	735
<code>Ipopt::SmartPtr< Ipopt::ConvergenceCheck ></code>	735
<code>Ipopt::SmartPtr< Ipopt::DenseGenMatrix ></code>	735
<code>Ipopt::SmartPtr< Ipopt::DenseSymMatrix ></code>	735
<code>Ipopt::SmartPtr< Ipopt::DenseVector ></code>	735
<code>Ipopt::SmartPtr< Ipopt::DiagMatrix ></code>	735
<code>Ipopt::SmartPtr< Ipopt::DiagMatrixSpace ></code>	735
<code>Ipopt::SmartPtr< Ipopt::EqMultiplierCalculator ></code>	735
<code>Ipopt::SmartPtr< Ipopt::ExpandedMultiVectorMatrix ></code>	735
<code>Ipopt::SmartPtr< Ipopt::ExpansionMatrix ></code>	735
<code>Ipopt::SmartPtr< Ipopt::ExpansionMatrixSpace ></code>	735
<code>Ipopt::SmartPtr< Ipopt::GenKKTsSolverInterface ></code>	735
<code>Ipopt::SmartPtr< Ipopt::HessianUpdater ></code>	735
<code>Ipopt::SmartPtr< Ipopt::IdentityMatrixSpace ></code>	735
<code>Ipopt::SmartPtr< Ipopt::InexactNewtonNormalStep ></code>	735
<code>Ipopt::SmartPtr< Ipopt::InexactNormalStepCalculator ></code>	735
<code>Ipopt::SmartPtr< Ipopt::InexactNormalTerminationTester ></code>	735

Ipopt::SmartPtr< Ipopt::InexactPDSolver >	735
Ipopt::SmartPtr< Ipopt::IpoptAdditionalCq >	735
Ipopt::SmartPtr< Ipopt::IpoptAdditionalData >	735
Ipopt::SmartPtr< Ipopt::IpoptAlgorithm >	735
Ipopt::SmartPtr< Ipopt::IpoptCalculatedQuantities >	735
Ipopt::SmartPtr< Ipopt::IpoptData >	735
Ipopt::SmartPtr< Ipopt::IpoptNLP >	735
Ipopt::SmartPtr< Ipopt::IterateInitializer >	735
Ipopt::SmartPtr< Ipopt::IteratesVectorSpace >	735
Ipopt::SmartPtr< Ipopt::IterationOutput >	735
Ipopt::SmartPtr< Ipopt::IterativeSolverTerminationTester >	735
Ipopt::SmartPtr< Ipopt::Journal >	735
Ipopt::SmartPtr< Ipopt::Journalist >	735
Ipopt::SmartPtr< Ipopt::LineSearch >	735
Ipopt::SmartPtr< Ipopt::Matrix >	735
Ipopt::SmartPtr< Ipopt::MultiVectorMatrix >	735
Ipopt::SmartPtr< Ipopt::MuOracle >	735
Ipopt::SmartPtr< Ipopt::MuUpdate >	735
Ipopt::SmartPtr< Ipopt::NLP >	735
Ipopt::SmartPtr< Ipopt::NLPScalingObject >	735
Ipopt::SmartPtr< Ipopt::OptionsList >	735
Ipopt::SmartPtr< Ipopt::OrigIterationOutput >	735
Ipopt::SmartPtr< Ipopt::PDPerturbationHandler >	735
Ipopt::SmartPtr< Ipopt::PDSystemSolver >	735
Ipopt::SmartPtr< Ipopt::RegisteredOption >	735
Ipopt::SmartPtr< Ipopt::RegisteredOptions >	735
Ipopt::SmartPtr< Ipopt::RestorationPhase >	735
Ipopt::SmartPtr< Ipopt::ScaledMatrixSpace >	735
Ipopt::SmartPtr< Ipopt::SearchDirectionCalculator >	735
Ipopt::SmartPtr< Ipopt::SolveStatistics >	735

Ipopt::SmartPtr< Ipopt::SparseSymLinearSolverInterface >	735
Ipopt::SmartPtr< Ipopt::SumSymMatrixSpace >	735
Ipopt::SmartPtr< Ipopt::SymLinearSolver >	735
Ipopt::SmartPtr< Ipopt::SymMatrix >	735
Ipopt::SmartPtr< Ipopt::SymScaledMatrixSpace >	735
Ipopt::SmartPtr< Ipopt::TDependencyDetector >	735
Ipopt::SmartPtr< Ipopt::TNLP >	735
Ipopt::SmartPtr< Ipopt::TripletToCSRConverter >	735
Ipopt::SmartPtr< Ipopt::TSymLinearSolver >	735
Ipopt::SmartPtr< Ipopt::TSymScalingMethod >	735
Ipopt::SmartPtr< Ipopt::Vector >	735
std::list< T >::reverse_iterator	
std::forward_list< T >::reverse_iterator	
std::unordered_multimap< K, T >::reverse_iterator	
std::unordered_map< K, T >::reverse_iterator	
std::multimap< K, T >::reverse_iterator	
std::set< K >::reverse_iterator	
std::string::reverse_iterator	
std::unordered_set< K >::reverse_iterator	
std::map< K, T >::reverse_iterator	
std::multiset< K >::reverse_iterator	
std::wstring::reverse_iterator	
std::unordered_multiset< K >::reverse_iterator	
std::basic_string< Char >::reverse_iterator	
std::deque< T >::reverse_iterator	
std::vector< T >::reverse_iterator	
std::set< K >	
std::smart_ptr< T >	
std::stack< T >	
Ipopt::RegisteredOption::string_entry	777
Ipopt::Subject	778
Ipopt::TaggedObject	813
std::system_error	
std::thread	
Ipopt::TimedTask	819
Ipopt::TripletToCSRConverter::TripletEntry	864
Ipopt::TripletHelper	866
std::unique_ptr< T >	
std::unordered_map< K, T >	
std::unordered_multimap< K, T >	

```

std::unordered_multiset< K >
std::unordered_set< K >
std::valarray< T >
std::vector< T >
std::vector< const Ipopt::Subject * >
std::vector< Index >
std::vector< Ipopt::Observer * >
std::vector< Ipopt::PiecewisePenEntry >
std::vector< Ipopt::RegisteredOption::string_entry >
std::vector< Ipopt::SmartPtr< const Ipopt::Matrix > >
std::vector< Ipopt::SmartPtr< const Ipopt::MatrixSpace > >
std::vector< Ipopt::SmartPtr< const Ipopt::SymMatrix > >
std::vector< Ipopt::SmartPtr< const Ipopt::SymMatrixSpace > >
std::vector< Ipopt::SmartPtr< const Ipopt::Vector > >
std::vector< Ipopt::SmartPtr< const Ipopt::VectorSpace > >
std::vector< Ipopt::SmartPtr< Ipopt::Journal > >
std::vector< Ipopt::SmartPtr< Ipopt::Vector > >
std::vector< Ipopt::TaggedObject::Tag >
std::vector< Number >
std::vector< std::string >
std::vector< std::vector< bool > >
std::vector< std::vector< Ipopt::SmartPtr< const Ipopt::Matrix > > >
std::vector< std::vector< Ipopt::SmartPtr< const Ipopt::MatrixSpace > > >
std::vector< std::vector< Ipopt::SmartPtr< Ipopt::Matrix > > >
std::vector< Suffix_Source >
std::vector< Suffix_Type >
std::weak_ptr< T >
K
T

```

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Ipopt::AdaptiveMuUpdate	
Non-monotone mu update	55
Ipopt::AlgorithmBuilder	
Builder to create a complete IpoptAlg object	63
Ipopt::AlgorithmStrategyObject	
This is the base class for all algorithm strategy objects	65
Ipopt::AmplOptionsList::AmplOption	
Ampl Option class, contains name, type and description for an AMPL option	70
Ipopt::AmplOptionsList	
Class for storing a number of AMPL options that should be registered to the AMPL Solver library interface	72
Ipopt::AmplSuffixHandler	74

Ipopt::AmplTNLP	
Ampl Interface	77
Ipopt::AugRestoSystemSolver	
Class that converts the an augmented system with compound restoration pieces into a smaller "pivoted" system to be solved with an existing AugSystemSolver	87
Ipopt::AugSystemSolver	
Base class for Solver for the augmented system	92
Ipopt::BacktrackingLineSearch	
General implementation of a backtracking line search	96
Ipopt::BacktrackingLSAcceptor	
Base class for backtracking line search acceptors	107
Ipopt::CachedResults< T >	
Cache Priority Enum	112
Ipopt::CGPenaltyCq	
Class for all Chen-Goldfarb penalty method specific calculated quantities	117
Ipopt::CGPenaltyData	
Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm	122
Ipopt::CGPenaltyLSAcceptor	
Line search acceptor, based on the Chen-Goldfarb penalty function approach	128
Ipopt::CGPerturbationHandler	
Class for handling the perturbation factors delta_x, delta_s, delta_c, and delta_d in the primal dual system	139
Ipopt::CGSearchDirCalculator	
Implementation of the search direction calculator that computes the Chen-Goldfarb step for the current barrier and penalty parameter	146
Ipopt::CompoundMatrix	
Class for Matrices consisting of other matrices	151
Ipopt::CompoundMatrixSpace	
This is the matrix space for CompoundMatrix	156
Ipopt::CompoundSymMatrix	
Class for symmetric matrices consisting of other matrices	160
Ipopt::CompoundSymMatrixSpace	
This is the matrix space for CompoundSymMatrix	165
Ipopt::CompoundVector	
Class of Vectors consisting of other vectors	168
Ipopt::CompoundVectorSpace	
This vectors space is the vector space for CompoundVector	176
Ipopt::ConvergenceCheck	
Base class for checking the algorithm termination criteria	180

Ipopt::DefaultIterateInitializer	
Class implementing the default initialization procedure (based on user options) for the iterates	182
Ipopt::DenseGenMatrix	
Class for dense general matrices	188
Ipopt::DenseGenMatrixSpace	
This is the matrix space for DenseGenMatrix	194
Ipopt::DenseSymMatrix	
Class for dense symetrix matrices	195
Ipopt::DenseSymMatrixSpace	
This is the matrix space for DenseSymMatrix	199
Ipopt::DenseVector	
Dense Vector Implementation	200
Ipopt::DenseVectorSpace	
This vectors space is the vector space for DenseVector	209
Ipopt::DependentResult< T >	
Templated class which stores one entry for the CachedResult class	213
Ipopt::DiagMatrix	
Class for diagonal matrices	216
Ipopt::DiagMatrixSpace	
This is the matrix space for DiagMatrix	219
Ipopt::EqMultiplierCalculator	
Base Class for objects that compute estimates for the equality constraint multipliers y_c and y_d	221
Ipopt::EquilibrationScaling	
This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point	223
Ipopt::ExactHessianUpdater	
Implementation of the HessianUpdater for the use of exact second derivatives	226
Ipopt::ExpandedMultiVectorMatrix	
Class for Matrices with few rows that consists of Vectors, together with a premultiplied Expansion matrix	227
Ipopt::ExpandedMultiVectorMatrixSpace	
This is the matrix space for ExpandedMultiVectorMatrix	231
Ipopt::ExpansionMatrix	
Class for expansion/projection matrices	233
Ipopt::ExpansionMatrixSpace	
This is the matrix space for ExpansionMatrix	237
Ipopt::FileJournal	
FileJournal class	240

Ipopt::Filter	
Class for the filter	242
Ipopt::FilterEntry	
Class for one filter entry	244
Ipopt::FilterLSAcceptor	
Filter line search	247
Ipopt::GenAugSystemSolver	
Solver for the augmented system using GenKKTSolverInterfaces	256
Ipopt::GenKKTSolverInterface	
Base class for interfaces to symmetric indefinite linear solvers for generic matrices	262
Ipopt::GenTMatrix	
Class for general matrices stored in triplet format	264
Ipopt::GenTMatrixSpace	
This is the matrix space for a GenTMatrix with fixed sparsity structure	269
Ipopt::GradientScaling	
This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point	272
Ipopt::HessianUpdater	
Abstract base class for objects responsible for updating the Hessian information	275
Ipopt::IdentityMatrix	
Class for Matrices which are multiples of the identity matrix	276
Ipopt::IdentityMatrixSpace	
This is the matrix space for IdentityMatrix	280
Ipopt::InexactAlgorithmBuilder	
Builder to create a complete IpoptAlg object for the inexact step computation version	281
Ipopt::InexactCq	
Class for all Chen-Goldfarb penalty method specific calculated quantities	284
Ipopt::InexactData	
Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm	289
Ipopt::InexactDoglegNormalStep	
Compute the normal step using a dogleg approach	294
Ipopt::InexactLSAcceptor	
Penalty function line search for the inexact step algorithm version	297
Ipopt::InexactNewtonNormalStep	
Compute the "Newton" normal step from the (slack-scaled) augmented system	305
Ipopt::InexactNormalStepCalculator	
Base class for computing the normal step for the inexact step calculation algorithm	308
Ipopt::InexactNormalTerminationTester	
This class implements the termination tests for the primal-dual system	310

<code>Ipopt::InexactPDSolver</code>	
This is the implemetation of the Primal-Dual System, allowing the usage of an inexact linear solver	313
<code>Ipopt::InexactPDTerminationTester</code>	
This class implements the termination tests for the primal-dual system	317
<code>Ipopt::InexactSearchDirCalculator</code>	
Implementation of the search direction calculator that computes the search direction using iterative linear solvers	323
<code>Ipopt::InexactTSymScalingMethod</code>	
Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm	327
<code>Ipopt::IpoptAdditionalCq</code>	
Base class for additional calculated quantities that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers	329
<code>Ipopt::IpoptAdditionalData</code>	
Base class for additional data that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers	330
<code>Ipopt::IpoptAlgorithm</code>	
The main ipopt algorithm class	332
<code>Ipopt::IpoptApplication</code>	
This is the main application class for making calls to <code>Ipopt</code>	338
<code>Ipopt::IpoptCalculatedQuantities</code>	
Class for all IPOPT specific calculated quantities	345
<code>Ipopt::IpoptData</code>	
Class to organize all the data required by the algorithm	374
<code>Ipopt::IpoptException</code>	
This is the base class for all exceptions	389
<code>Ipopt::IpoptNLP</code>	
This is the abstract base class for classes that map the traditional NLP into something that is more useful by <code>Ipopt</code>	391
<code>Ipopt::IterateInitializer</code>	
Base class for all methods for initializing the iterates	399
<code>Ipopt::IteratesVector</code>	
Specialized CompoundVector class specifically for the algorithm iterates	400
<code>Ipopt::IteratesVectorSpace</code>	
Vector Space for the IteratesVector class	412
<code>Ipopt::IterationOutput</code>	
Base class for objects that do the output summary per iteration	415
<code>Ipopt::IterativePardisoSolverInterface</code>	
Interface to the linear solver Pardiso, derived from SparseSymLinearSolverInterface	417

Ipopt::IterativeSolverTerminationTester	
This base class is for the termination tests for the iterative linear solver in the inexact version of Ipopt	427
Ipopt::IterativeWsmvSolverInterface	
Interface to the linear solver WISMP, derived from SparseSymLinearSolverInterface	430
Ipopt::Journal	
Journal class (part of the Journalist implementation.)	436
Ipopt::Journalist	
Class responsible for all message output	440
Ipopt::LeastSquareMultipliers	
Class for calculator for the least-square equality constraint multipliers	444
Ipopt::LimMemQuasiNewtonUpdater	
Implementation of the HessianUpdater for limit-memory quasi-Newton approximation of the Lagrangian Hessian	446
Ipopt::LineSearch	
Base class for line search objects	459
Ipopt::LoqoMuOracle	
Implementation of the LOQO formula for computing the barrier parameter	461
Ipopt::LowRankAugSystemSolver	
Solver for the augmented system with LowRankUpdateSymMatrix Hessian matrices	463
Ipopt::LowRankSSAugSystemSolver	
Solver for the augmented system with LowRankUpdateSymMatrix Hessian matrices	470
Ipopt::LowRankUpdateSymMatrix	
Class for symmetric matrices, represented as low-rank updates	476
Ipopt::LowRankUpdateSymMatrixSpace	
This is the matrix space for LowRankUpdateSymMatrix	481
Ipopt::Ma27TSolverInterface	
Interface to the symmetric linear solver MA27, derived from SparseSymLinearSolverInterface	484
Ipopt::Ma28TDependencyDetector	
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian	491
Ipopt::Ma57TSolverInterface	
Interface to the symmetric linear solver MA57, derived from SparseSymLinearSolverInterface	494
ma77_control_d	500
ma77_info_d	503
Ipopt::Ma77SolverInterface	
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices	507
ma86_control_d	512

ma86_info_d	514
Ipopt::Ma86SolverInterface	
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices	515
ma97_control_d	521
ma97_info	523
Ipopt::Ma97SolverInterface	
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices	525
Ipopt::Matrix	
Matrix Base Class	532
Ipopt::MatrixSpace	
MatrixSpace base class, corresponding to the Matrix base class	539
Ipopt::Mc19TSymScalingMethod	
Class for the method for computing scaling factors for symmetric matrices in triplet format, using MC19	542
mc68_control	543
mc68_info	545
Ipopt::MinC_1NrmRestorationPhase	
Restoration Phase that minimizes the 1-norm of the constraint violation - using the interior point method (Ipopt)	546
Ipopt::MonotoneMuUpdate	
Monotone Mu Update	550
Ipopt::MultiVectorMatrix	
Class for Matrices with few columns that consists of Vectors	553
Ipopt::MultiVectorMatrixSpace	
This is the matrix space for MultiVectorMatrix	558
Ipopt::MumpsSolverInterface	
Interface to the linear solver Mumps, derived from SparseSymLinearSolverInterface	560
Ipopt::MuOracle	
Abstract Base Class for classes that are able to compute a suggested value of the barrier parameter that can be used as an oracle in the NonmontoneMuUpdate class	566
Ipopt::MuUpdate	
Abstract Base Class for classes that implement methods for computing the barrier and fraction-to-the-boundary rule parameter for the current iteration	568
Ipopt::NLP	
Brief Class Description	570
Ipopt::NLPBoundsRemover	
This is an adapter for an NLP that converts variable bound constraints to inequality constraints	575

Ipopt::NLPScalingObject	
This is the abstract base class for problem scaling	581
Ipopt::NoNLPScalingObject	
Class implementing the scaling object that doesn't to any scaling	588
Ipopt::Observer	
Slight Variation of the Observer Design Pattern	590
Ipopt::OptimalityErrorConvergenceCheck	
Brief Class Description	593
Ipopt::OptionsList	
This class stores a list of user set options	597
Ipopt::OptionsList::OptionValue	
Class for storing the value and counter for each option in OptionsList	602
Ipopt::OrigIpoptNLP	
This class maps the traditional NLP into something that is more useful by Ipopt	605
Ipopt::OrigIterationOutput	
Class for the iteration summary output for the original NLP	621
Ipopt::PardisoSolverInterface	
Interface to the linear solver Pardiso, derived from SparseSymLinearSolverInterface	623
Ipopt::PDFullSpaceSolver	
This is the implemetation of the Primal-Dual System, using the full space approach with a direct linear solver	631
Ipopt::PDPerturbationHandler	
Class for handling the perturbation factors delta_x, delta_s, delta_c, and delta_d in the primal dual system	635
Ipopt::PDSearchDirCalculator	
Implementation of the search direction calculator that computes the pure primal dual step for the current barrier parameter	643
Ipopt::PDSysstemSolver	
Pure Primal Dual System Solver Base Class	645
Ipopt::PenaltyLSAcceptor	
Penalty function line search	648
Ipopt::PiecewisePenalty	
Class for the Piecewise Penalty	655
Ipopt::PiecewisePenEntry	
Struct for one Piecewise Penalty entry	658
Ipopt::PointPerturber	
This class is a simple object for generating randomly perturbed points that are withing the NLP bounds	659
Ipopt::AmplOptionsList::PrivatInfo	660

<code>Ipopt::ProbingMuOracle</code>	
Implementation of the probing strategy for computing the barrier parameter	662
<code>Ipopt::QualityFunctionMuOracle</code>	
Implementation of the probing strategy for computing the barrier parameter	664
<code>Ipopt::ReferencedObject</code>	
<code>ReferencedObject</code> class	673
<code>Ipopt::Referencer</code>	
Psydo-class, from which everything has to inherit that wants to use be registered as a <code>Referencer</code> for a <code>ReferencedObject</code>	677
<code>Ipopt::RegisteredOption</code>	
Base class for registered options	678
<code>Ipopt::RegisteredOptions</code>	
Class for storing registered options	687
<code>Ipopt::RestoConvergenceCheck</code>	
Convergence check for the restoration phase	693
<code>Ipopt::RestoFilterConvergenceCheck</code>	
This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism	696
<code>Ipopt::RestoIpoptNLP</code>	
This class maps the traditional <code>NLP</code> into something that is more useful by <code>Ipopt</code>	699
<code>Ipopt::RestoIterateInitializer</code>	
Class implementing the default initialization procedure (based on user options) for the iterates	712
<code>Ipopt::RestoIterationOutput</code>	
Class for the iteration summary output for the restoration phase	715
<code>Ipopt::RestoPenaltyConvergenceCheck</code>	
This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism	718
<code>Ipopt::RestorationPhase</code>	
Base class for different restoration phases	721
<code>Ipopt::RestoRestorationPhase</code>	
Recursive Restoration Phase for the <code>MinC_1NrmRestorationPhase</code>	722
<code>Ipopt::ScaledMatrix</code>	
Class for a <code>Matrix</code> in conjunction with its scaling factors for row and column scaling	724
<code>Ipopt::ScaledMatrixSpace</code>	
This is the matrix space for <code>ScaledMatrix</code>	729
<code>Ipopt::SearchDirectionCalculator</code>	
Base class for computing the search direction for the line search	732
<code>Ipopt::SlackBasedTSymScalingMethod</code>	
Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the <code>inexaxct</code> algorithm	733

Ipopt::SmartPtr< T >	
Template class for Smart Pointers	735
Ipopt::SolveStatistics	
This class collects statistics about an optimziation run, such as iteration count, final infeasibilities etc	742
Ipopt::SparseSymLinearSolverInterface	
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices	747
Ipopt::StandardScalingBase	
This is a base class for many standard scaling techniques	753
Ipopt::StdAugSystemSolver	
Solver for the augmented system for triple type matrices	759
Ipopt::StdInterfaceTNLP	
Implementation of a TNLP for the Standard C interface	766
Ipopt::StreamJournal	
StreamJournal class	775
Ipopt::RegisteredOption::string_entry	
Class to hold the valid string settings for a string option	777
Ipopt::Subject	
Slight Variation of the Observer Design Pattern (Subject part)	778
Ipopt::SumMatrix	
Class for Matrices which are sum of matrices	781
Ipopt::SumMatrixSpace	
Class for matrix space for SumMatrix	784
Ipopt::SumSymMatrix	
Class for Matrices which are sum of symmetric matrices	787
Ipopt::SumSymMatrixSpace	
Class for matrix space for SumSymMatrix	790
Ipopt::SymLinearSolver	
Base class for all derived symmetric linear solvers	792
Ipopt::SymMatrix	
This is the base class for all derived symmetric matrix types	795
Ipopt::SymMatrixSpace	
SymMatrixSpace base class, corresponding to the SymMatrix base class	797
Ipopt::SymScaledMatrix	
Class for a Matrix in conjunction with its scaling factors for row and column scaling	800
Ipopt::SymScaledMatrixSpace	
This is the matrix space for SymScaledMatrix	803
Ipopt::SymTMatrix	
Class for symmetric matrices stored in triplet format	806

Ipopt::SymTMatrixSpace	
This is the matrix space for a SymTMatrix with fixed sparsity structure	811
Ipopt::TaggedObject	
TaggedObject class	813
Ipopt::TDependencyDetector	
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian	817
Ipopt::TimedTask	
This class is used to collect timing information for a particular task	819
Ipopt::TimingStatistics	
This class collects all timing statistics for Ipopt	822
Ipopt::TNLP	
Base class for all NLP 's that use standard triplet matrix form and dense vectors	829
Ipopt::TNLPAdapter	
This class Adapts the TNLP interface so it looks like an NLP interface	836
Ipopt::TNLPReducer	
This is a wrapper around a given TNLP class that takes out a list of constraints that are given to the constructor	852
Ipopt::TransposeMatrix	
Class for Matrices which are the transpose of another matrix	859
Ipopt::TransposeMatrixSpace	
This is the matrix space for TransposeMatrix	862
Ipopt::TripletToCSRConverter::TripletEntry	
Class for one triplet position entry	864
Ipopt::TripletHelper	866
Ipopt::TripletToCSRConverter	
Class for converting symmetric matrices given in triplet format to matrices in compressed sparse row (CSR) format of the upper triangular part (or, equivalently, compressed sparse column (CSC) format for the lower triangular part)	870
Ipopt::TSymDependencyDetector	
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian	875
Ipopt::TSymLinearSolver	
General driver for linear solvers for sparse indefinite symmetric matrices	877
Ipopt::TSymScalingMethod	
Base class for the method for computing scaling factors for symmetric matrices in triplet format	883
Ipopt::UserScaling	
This class does problem scaling by getting scaling parameters from the user (through the NLP interface)	885

Ipopt::Vector Vector Base Class	887
Ipopt::VectorSpace VectorSpace base class, corresponding to the Vector base class	899
Ipopt::WarmStartIterateInitializer Class implementing an initialization procedure for warm starts	901
Ipopt::WsmvSolverInterface Interface to the linear solver Wsmv, derived from SparseSymLinearSolverInterface	905
Ipopt::ZeroMatrix Class for Matrices with only zero entries	912
Ipopt::ZeroMatrixSpace Class for matrix space for ZeroMatrix	914
Ipopt::ZeroSymMatrix Class for Symmetric Matrices with only zero entries	916
Ipopt::ZeroSymMatrixSpace Class for matrix space for ZeroSymMatrix	919

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

Algorithm/IpAdaptiveMuUpdate.hpp	926
Algorithm/IpAlgBuilder.hpp	926
Algorithm/IpAlgorithmRegOp.hpp	927
Algorithm/IpAlgStrategy.hpp	927
Algorithm/IpAugRestoSystemSolver.hpp	927
Algorithm/IpAugSystemSolver.hpp	928
Algorithm/IpBacktrackingLineSearch.hpp	928
Algorithm/IpBacktrackingLSAccepter.hpp	929
Algorithm/IpConvCheck.hpp	929
Algorithm/IpDefaultIterateInitializer.hpp	929
Algorithm/IpEqMultCalculator.hpp	930
Algorithm/IpEquilibrationScaling.hpp	930
Algorithm/IpExactHessianUpdater.hpp	930

Algorithm/ IpFilter.hpp	931
Algorithm/ IpFilterLSAcceptor.hpp	931
Algorithm/ IpGenAugSystemSolver.hpp	931
Algorithm/ IpGradientScaling.hpp	932
Algorithm/ IpHessianUpdater.hpp	932
Algorithm/ IpIpoptAlg.hpp	932
Algorithm/ IpIpoptCalculatedQuantities.hpp	933
Algorithm/ IpIpoptData.hpp	933
Algorithm/ IpIpoptNLP.hpp	934
Algorithm/ IpIterateInitializer.hpp	934
Algorithm/ IpIteratesVector.hpp	935
Algorithm/ IpIterationOutput.hpp	935
Algorithm/ IpLeastSquareMults.hpp	935
Algorithm/ IpLimMemQuasiNewtonUpdater.hpp	936
Algorithm/ IpLineSearch.hpp	936
Algorithm/ IpLoqoMuOracle.hpp	936
Algorithm/ IpLowRankAugSystemSolver.hpp	937
Algorithm/ IpLowRankSSAugSystemSolver.hpp	937
Algorithm/ IpMonotoneMuUpdate.hpp	937
Algorithm/ IpMuOracle.hpp	938
Algorithm/ IpMuUpdate.hpp	938
Algorithm/ IpNLPBoundsRemover.hpp	938
Algorithm/ IpNLPScaling.hpp	939
Algorithm/ IpOptErrorConvCheck.hpp	939
Algorithm/ IpOrigIpoptNLP.hpp	939
Algorithm/ IpOrigIterationOutput.hpp	940
Algorithm/ IpPDFullSpaceSolver.hpp	940
Algorithm/ IpPDPerturbationHandler.hpp	940
Algorithm/ IpPDSearchDirCalc.hpp	941
Algorithm/ IpPDSysSystemSolver.hpp	941

Algorithm/ IpPenaltyLSAcceptor.hpp	941
Algorithm/ IpProbingMuOracle.hpp	942
Algorithm/ IpQualityFunctionMuOracle.hpp	942
Algorithm/ IpRestoConvCheck.hpp	942
Algorithm/ IpRestoFilterConvCheck.hpp	943
Algorithm/ IpRestoIpoptNLP.hpp	943
Algorithm/ IpRestoIterateInitializer.hpp	943
Algorithm/ IpRestoIterationOutput.hpp	944
Algorithm/ IpRestoMinC_1Nrm.hpp	944
Algorithm/ IpRestoPenaltyConvCheck.hpp	944
Algorithm/ IpRestoPhase.hpp	945
Algorithm/ IpRestoRestoPhase.hpp	945
Algorithm/ IpSearchDirCalculator.hpp	946
Algorithm/ IpStdAugSystemSolver.hpp	946
Algorithm/ IpTimingStatistics.hpp	946
Algorithm/ IpUserScaling.hpp	947
Algorithm/ IpWarmStartIterateInitializer.hpp	947
Algorithm/Inexact/ IpInexactAlgBuilder.hpp	921
Algorithm/Inexact/ IpInexactCq.hpp	921
Algorithm/Inexact/ IpInexactData.hpp	922
Algorithm/Inexact/ IpInexactDoglegNormal.hpp	922
Algorithm/Inexact/ IpInexactLSAcceptor.hpp	922
Algorithm/Inexact/ IpInexactNewtonNormal.hpp	923
Algorithm/Inexact/ IpInexactNormalStepCalc.hpp	923
Algorithm/Inexact/ IpInexactNormalTerminationTester.hpp	923
Algorithm/Inexact/ IpInexactPDSolver.hpp	924
Algorithm/Inexact/ IpInexactPDTerminationTester.hpp	924
Algorithm/Inexact/ IpInexactRegOp.hpp	924
Algorithm/Inexact/ IpInexactSearchDirCalc.hpp	925
Algorithm/Inexact/ IpInexactTSymScalingMethod.hpp	925

Algorithm/Inexact/ IpIterativePardisoSolverInterface.hpp	925
Algorithm/Inexact/ IpIterativeSolverTerminationTester.hpp	926
Algorithm/LinearSolvers/ hsl_ma77d.h	947
Algorithm/LinearSolvers/ hsl_ma86d.h	951
Algorithm/LinearSolvers/ hsl_ma97d.h	953
Algorithm/LinearSolvers/ hsl_mc68i.h	957
Algorithm/LinearSolvers/ IpGenKKTsSolverInterface.hpp	958
Algorithm/LinearSolvers/ IpIterativeWsmvSolverInterface.hpp	958
Algorithm/LinearSolvers/ IpLinearSolversRegOp.hpp	958
Algorithm/LinearSolvers/ IpMa27TSolverInterface.hpp	959
Algorithm/LinearSolvers/ IpMa28TDependencyDetector.hpp	959
Algorithm/LinearSolvers/ IpMa57TSolverInterface.hpp	959
Algorithm/LinearSolvers/ IpMa77SolverInterface.hpp	960
Algorithm/LinearSolvers/ IpMa86SolverInterface.hpp	960
Algorithm/LinearSolvers/ IpMa97SolverInterface.hpp	960
Algorithm/LinearSolvers/ IpMc19TSymScalingMethod.hpp	961
Algorithm/LinearSolvers/ IpMumpsSolverInterface.hpp	961
Algorithm/LinearSolvers/ IpPardisoSolverInterface.hpp	961
Algorithm/LinearSolvers/ IpSlackBasedTSymScalingMethod.hpp	962
Algorithm/LinearSolvers/ IpSparseSymLinearSolverInterface.hpp	962
Algorithm/LinearSolvers/ IpSymLinearSolver.hpp	962
Algorithm/LinearSolvers/ IpTDependencyDetector.hpp	963
Algorithm/LinearSolvers/ IpTripletToCSRConverter.hpp	963
Algorithm/LinearSolvers/ IpTSymDependencyDetector.hpp	964
Algorithm/LinearSolvers/ IpTSymLinearSolver.hpp	964
Algorithm/LinearSolvers/ IpTSymScalingMethod.hpp	964
Algorithm/LinearSolvers/ IpWsmvSolverInterface.hpp	965
Apps/AmplSolver/ AmplTNLP.hpp	965
Common/ config_default.h	966
Common/ config_ipopt_default.h	966

Common/ IpCachedResults.hpp	967
Common/ IpDebug.hpp	967
Common/ IpException.hpp	969
Common/ IpJournalist.hpp	970
Common/ IpObserver.hpp	971
Common/ IpoptConfig.h	971
Common/ IpOptionsList.hpp	971
Common/ IpReferenced.hpp	972
Common/ IpRegOptions.hpp	972
Common/ IpSmartPtr.hpp	973
Common/ IpTaggedObject.hpp	974
Common/ IpTimedTask.hpp	975
Common/ IpTypes.hpp	975
Common/ IpUtils.hpp	976
contrib/CGPenalty/ IpCGPenaltyCq.hpp	976
contrib/CGPenalty/ IpCGPenaltyData.hpp	977
contrib/CGPenalty/ IpCGPenaltyLSAcceptor.hpp	977
contrib/CGPenalty/ IpCGPenaltyRegOp.hpp	977
contrib/CGPenalty/ IpCGPerturbationHandler.hpp	978
contrib/CGPenalty/ IpCGSearchDirCalc.hpp	978
contrib/CGPenalty/ IpPiecewisePenalty.hpp	979
contrib/LinearSolverLoader/ HSLLoader.h	979
contrib/LinearSolverLoader/ LibraryHandler.h	992
contrib/LinearSolverLoader/ PardisoLoader.h	993
Interfaces/ IpAlgTypes.hpp	994
Interfaces/ IpInterfacesRegOp.hpp	995
Interfaces/ IpIpoptApplication.hpp	995
Interfaces/ IpNLP.hpp	996
Interfaces/ IpReturnCodes.h	996
Interfaces/ IpReturnCodes.hpp	996

Interfaces/ lpReturnCodes_inc.h	997
Interfaces/ lpSolveStatistics.hpp	998
Interfaces/ lpStdCInterface.h	998
Interfaces/ lpStdInterfaceTNLP.hpp	1007
Interfaces/ lpTNLP.hpp	1007
Interfaces/ lpTNLPAdapter.hpp	1008
Interfaces/ lpTNLPReducer.hpp	1008
LinAlg/ lpBlas.hpp	1008
LinAlg/ lpCompoundMatrix.hpp	1009
LinAlg/ lpCompoundSymMatrix.hpp	1009
LinAlg/ lpCompoundVector.hpp	1010
LinAlg/ lpDenseGenMatrix.hpp	1010
LinAlg/ lpDenseSymMatrix.hpp	1011
LinAlg/ lpDenseVector.hpp	1011
LinAlg/ lpDiagMatrix.hpp	1012
LinAlg/ lpExpandedMultiVectorMatrix.hpp	1012
LinAlg/ lpExpansionMatrix.hpp	1012
LinAlg/ lpIdentityMatrix.hpp	1013
LinAlg/ lpLapack.hpp	1013
LinAlg/ lpLowRankUpdateSymMatrix.hpp	1014
LinAlg/ lpMatrix.hpp	1014
LinAlg/ lpMultiVectorMatrix.hpp	1015
LinAlg/ lpScaledMatrix.hpp	1015
LinAlg/ lpSumMatrix.hpp	1015
LinAlg/ lpSumSymMatrix.hpp	1016
LinAlg/ lpSymMatrix.hpp	1016
LinAlg/ lpSymScaledMatrix.hpp	1017
LinAlg/ lpTransposeMatrix.hpp	1017
LinAlg/ lpVector.hpp	1017
LinAlg/ lpZeroMatrix.hpp	1018

LinAlg/lpZeroSymMatrix.hpp	1018
LinAlg/TMatrices/lpGenTMatrix.hpp	1019
LinAlg/TMatrices/lpSymTMatrix.hpp	1019
LinAlg/TMatrices/lpTripletHelper.hpp	1020

5 Namespace Documentation

5.1 Ipopt Namespace Reference

Classes

- class [InexactAlgorithmBuilder](#)
Builder to create a complete IpoptAlg object for the inexact step computation version.
- class [InexactCq](#)
Class for all Chen-Goldfarb penalty method specific calculated quantities.
- class [InexactData](#)
Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm.
- class [InexactDoglegNormalStep](#)
Compute the normal step using a dogleg approach.
- class [InexactLSAcceptor](#)
Penalty function line search for the inexact step algorithm version.
- class [InexactNewtonNormalStep](#)
Compute the "Newton" normal step from the (slack-scaled) augmented system.
- class [InexactNormalStepCalculator](#)
Base class for computing the normal step for the inexact step calculation algorithm.
- class [InexactNormalTerminationTester](#)
This class implements the termination tests for the primal-dual system.
- class [InexactPDSolver](#)
This is the implementation of the Primal-Dual System, allowing the usage of an inexact linear solver.
- class [InexactPDTerminationTester](#)
This class implements the termination tests for the primal-dual system.
- class [InexactSearchDirCalculator](#)
Implementation of the search direction calculator that computes the search direction using iterative linear solvers.
- class [InexactTSymScalingMethod](#)
Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm.
- class [IterativePardisoSolverInterface](#)
Interface to the linear solver Pardiso, derived from [SparseSymLinearSolverInterface](#).
- class [IterativeSolverTerminationTester](#)
This base class is for the termination tests for the iterative linear solver in the inexact version of [Ipopt](#).
- class [AdaptiveMuUpdate](#)
Non-monotone mu update.
- class [AlgorithmBuilder](#)
Builder to create a complete IpoptAlg object.
- class [AlgorithmStrategyObject](#)

- This is the base class for all algorithm strategy objects.*
- class [AugRestoSystemSolver](#)
Class that converts the an augmented system with compound restoration pieces into a smaller "pivoted" system to be solved with an existing [AugSystemSolver](#).
 - class [AugSystemSolver](#)
Base class for Solver for the augmented system.
 - class [BacktrackingLineSearch](#)
General implementation of a backtracking line search.
 - class [BacktrackingLSAcceptor](#)
Base class for backtracking line search acceptors.
 - class [ConvergenceCheck](#)
Base class for checking the algorithm termination criteria.
 - class [DefaultIterateInitializer](#)
Class implementing the default initialization procedure (based on user options) for the iterates.
 - class [EqMultiplierCalculator](#)
Base Class for objects that compute estimates for the equality constraint multipliers y_c and y_d .
 - class [EquilibrationScaling](#)
This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point.
 - class [PointPerturber](#)
This class is a simple object for generating randomly perturbed points that are withing the [NLP](#) bounds.
 - class [ExactHessianUpdater](#)
Implementation of the [HessianUpdater](#) for the use of exact second derivatives.
 - class [FilterEntry](#)
Class for one filter entry.
 - class [Filter](#)
Class for the filter.
 - class [FilterLSAcceptor](#)
[Filter](#) line search.
 - class [GenAugSystemSolver](#)
Solver for the augmented system using [GenKKTSolverInterfaces](#).
 - class [GradientScaling](#)
This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point.
 - class [HessianUpdater](#)
Abstract base class for objects responsible for updating the Hessian information.
 - class [IpoptAlgorithm](#)
The main ipopt algorithm class.
 - class [IpoptAdditionalCq](#)
Base class for additional calculated quantities that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers.
 - class [IpoptCalculatedQuantities](#)
Class for all IPOPT specific calculated quantities.
 - class [IpoptAdditionalData](#)
Base class for additional data that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers.
 - class [IpoptData](#)
Class to organize all the data required by the algorithm.

- class [IpoptNLP](#)
This is the abstract base class for classes that map the traditional [NLP](#) into something that is more useful by [Ipopt](#).
- class [IterateInitializer](#)
Base class for all methods for initializing the iterates.
- class [IteratesVector](#)
Specialized [CompoundVector](#) class specifically for the algorithm iterates.
- class [IteratesVectorSpace](#)
[Vector](#) Space for the [IteratesVector](#) class.
- class [IterationOutput](#)
Base class for objects that do the output summary per iteration.
- class [LeastSquareMultipliers](#)
Class for calculator for the least-square equality constraint multipliers.
- class [LimMemQuasiNewtonUpdater](#)
Implementation of the [HessianUpdater](#) for limit-memory quasi-Newton approximation of the Lagrangian Hessian.
- class [LineSearch](#)
Base class for line search objects.
- class [LoqoMuOracle](#)
Implementation of the LOQO formula for computing the barrier parameter.
- class [LowRankAugSystemSolver](#)
Solver for the augmented system with [LowRankUpdateSymMatrix](#) Hessian matrices.
- class [LowRankSSAugSystemSolver](#)
Solver for the augmented system with [LowRankUpdateSymMatrix](#) Hessian matrices.
- class [MonotoneMuUpdate](#)
Monotone Mu Update.
- class [MuOracle](#)
Abstract Base Class for classes that are able to compute a suggested value of the barrier parameter that can be used as an oracle in the [NonmontoneMuUpdate](#) class.
- class [MuUpdate](#)
Abstract Base Class for classes that implement methods for computing the barrier and fraction-to-the-boundary rule parameter for the current iteration.
- class [NLPBoundsRemover](#)
This is an adapter for an [NLP](#) that converts variable bound constraints to inequality constraints.
- class [NLPScalingObject](#)
This is the abstract base class for problem scaling.
- class [StandardScalingBase](#)
This is a base class for many standard scaling techniques.
- class [NoNLPScalingObject](#)
Class implementing the scaling object that doesn't to any scaling.
- class [OptimalityErrorConvergenceCheck](#)
Brief Class Description.
- class [OrigIpoptNLP](#)
This class maps the traditional [NLP](#) into something that is more useful by [Ipopt](#).
- class [OrigIterationOutput](#)
Class for the iteration summary output for the original [NLP](#).
- class [PDFullSpaceSolver](#)
This is the implemetation of the Primal-Dual System, using the full space approach with a direct linear solver.
- class [PDPerturbationHandler](#)

- Class for handling the perturbation factors δ_x , δ_s , δ_c , and δ_d in the primal dual system.*

 - class [PDSearchDirCalculator](#)

Implementation of the search direction calculator that computes the pure primal dual step for the current barrier parameter.
 - class [PDSolver](#)

Pure Primal Dual System Solver Base Class.
 - class [PenaltyLSAccepter](#)

Penalty function line search.
 - class [ProbingMuOracle](#)

Implementation of the probing strategy for computing the barrier parameter.
 - class [QualityFunctionMuOracle](#)

Implementation of the probing strategy for computing the barrier parameter.
 - class [RestoConvergenceCheck](#)

Convergence check for the restoration phase.
 - class [RestoFilterConvergenceCheck](#)

This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism.
 - class [RestoIpoptNLP](#)

This class maps the traditional [NLP](#) into something that is more useful by [Ipopt](#).
 - class [RestoIterateInitializer](#)

Class implementing the default initialization procedure (based on user options) for the iterates.
 - class [RestoIterationOutput](#)

Class for the iteration summary output for the restoration phase.
 - class [MinC_1NrmRestorationPhase](#)

Restoration Phase that minimizes the 1-norm of the constraint violation - using the interior point method ([Ipopt](#)).
 - class [RestoPenaltyConvergenceCheck](#)

This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism.
 - class [RestorationPhase](#)

Base class for different restoration phases.
 - class [RestoRestorationPhase](#)

Recursive Restoration Phase for the [MinC_1NrmRestorationPhase](#).
 - class [SearchDirectionCalculator](#)

Base class for computing the search direction for the line search.
 - class [StdAugSystemSolver](#)

Solver for the augmented system for triple type matrices.
 - class [TimingStatistics](#)

This class collects all timing statistics for [Ipopt](#).
 - class [UserScaling](#)

This class does problem scaling by getting scaling parameters from the user (through the [NLP](#) interface).
 - class [WarmStartIterateInitializer](#)

Class implementing an initialization procedure for warm starts.
 - class [GenKKTSolverInterface](#)

Base class for interfaces to symmetric indefinite linear solvers for generic matrices.
 - class [IterativeWsmvSolverInterface](#)

Interface to the linear solver WISMP, derived from [SparseSymLinearSolverInterface](#).
 - class [Ma27TSolverInterface](#)

Interface to the symmetric linear solver MA27, derived from [SparseSymLinearSolverInterface](#).

- class [Ma28TDependencyDetector](#)
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.
- class [Ma57TSolverInterface](#)
Interface to the symmetric linear solver MA57, derived from [SparseSymLinearSolverInterface](#).
- class [Ma77SolverInterface](#)
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.
- class [Ma86SolverInterface](#)
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.
- class [Ma97SolverInterface](#)
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.
- class [Mc19TSymScalingMethod](#)
Class for the method for computing scaling factors for symmetric matrices in triplet format, using MC19.
- class [MumpsSolverInterface](#)
Interface to the linear solver Mumps, derived from [SparseSymLinearSolverInterface](#).
- class [PardisoSolverInterface](#)
Interface to the linear solver Pardiso, derived from [SparseSymLinearSolverInterface](#).
- class [SlackBasedTSymScalingMethod](#)
Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm.
- class [SparseSymLinearSolverInterface](#)
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.
- class [SymLinearSolver](#)
Base class for all derived symmetric linear solvers.
- class [TDependencyDetector](#)
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.
- class [TripletToCSRConverter](#)
Class for converting symmetric matrices given in triplet format to matrices in compressed sparse row (CSR) format of the upper triangular part (or, equivalently, compressed sparse column (CSC) format for the lower triangular part).
- class [TSymDependencyDetector](#)
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.
- class [TSymLinearSolver](#)
General driver for linear solvers for sparse indefinite symmetric matrices.
- class [TSymScalingMethod](#)
Base class for the method for computing scaling factors for symmetric matrices in triplet format.
- class [WsmpSolverInterface](#)
Interface to the linear solver Wsmp, derived from [SparseSymLinearSolverInterface](#).
- class [AmplSuffixHandler](#)
- class [AmplOptionsList](#)
Class for storing a number of AMPL options that should be registered to the AMPL Solver library interface.
- class [AmplTNLP](#)
Ampl Interface.
- class [DependentResult](#)
Templated class which stores one entry for the [CachedResult](#) class.
- class [CachedResults](#)
Cache Priority Enum.
- class [IpoptException](#)
This is the base class for all exceptions.

- class [Journalist](#)
Class responsible for all message output.
- class [Journal](#)
Journal class (part of the [Journalist](#) implementation.).
- class [FileJournal](#)
FileJournal class.
- class [StreamJournal](#)
StreamJournal class.
- class [Observer](#)
Slight Variation of the [Observer](#) Design Pattern.
- class [Subject](#)
Slight Variation of the [Observer](#) Design Pattern ([Subject](#) part).
- class [OptionsList](#)
This class stores a list of user set options.
- class [Referencer](#)
Psydo-class, from which everything has to inherit that wants to use be registered as a [Referencer](#) for a [ReferencedObject](#).
- class [ReferencedObject](#)
ReferencedObject class.
- class [RegisteredOption](#)
Base class for registered options.
- class [RegisteredOptions](#)
Class for storing registered options.
- class [SmartPtr](#)
Template class for Smart Pointers.
- class [TaggedObject](#)
TaggedObject class.
- class [TimedTask](#)
This class is used to collect timing information for a particular task.
- class [CGPenaltyCq](#)
Class for all Chen-Goldfarb penalty method specific calculated quantities.
- class [CGPenaltyData](#)
Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm.
- class [CGPenaltyLSAcceptor](#)
Line search acceptor, based on the Chen-Goldfarb penalty function approach.
- class [CGPerturbationHandler](#)
Class for handling the perturbation factors δ_x , δ_s , δ_c , and δ_d in the primal dual system.
- class [CGSearchDirCalculator](#)
Implementation of the search direction calculator that computes the Chen-Goldfarb step for the current barrier and penalty parameter.
- struct [PiecewisePenEntry](#)
struct for one Piecewise Penalty entry.
- class [PiecewisePenalty](#)
Class for the Piecewise Penalty.
- class [IpoptApplication](#)
This is the main application class for making calls to [Ipopt](#).
- class [NLP](#)
Brief Class Description.

- class [SolveStatistics](#)
This class collects statistics about an optimization run, such as iteration count, final infeasibilities etc.
- class [StdInterfaceTNLP](#)
Implementation of a [TNLP](#) for the Standard C interface.
- class [TNLP](#)
Base class for all [NLP](#)'s that use standard triplet matrix form and dense vectors.
- class [TNLPAdapter](#)
This class Adapts the [TNLP](#) interface so it looks like an [NLP](#) interface.
- class [TNLPReducer](#)
This is a wrapper around a given [TNLP](#) class that takes out a list of constraints that are given to the constructor.
- class [CompoundMatrix](#)
Class for Matrices consisting of other matrices.
- class [CompoundMatrixSpace](#)
This is the matrix space for [CompoundMatrix](#).
- class [CompoundSymMatrix](#)
Class for symmetric matrices consisting of other matrices.
- class [CompoundSymMatrixSpace](#)
This is the matrix space for [CompoundSymMatrix](#).
- class [CompoundVector](#)
Class of Vectors consisting of other vectors.
- class [CompoundVectorSpace](#)
This vectors space is the vector space for [CompoundVector](#).
- class [DenseGenMatrix](#)
Class for dense general matrices.
- class [DenseGenMatrixSpace](#)
This is the matrix space for [DenseGenMatrix](#).
- class [DenseSymMatrix](#)
Class for dense symetrix matrices.
- class [DenseSymMatrixSpace](#)
This is the matrix space for [DenseSymMatrix](#).
- class [DenseVector](#)
Dense [Vector](#) Implementation.
- class [DenseVectorSpace](#)
This vectors space is the vector space for [DenseVector](#).
- class [DiagMatrix](#)
Class for diagonal matrices.
- class [DiagMatrixSpace](#)
This is the matrix space for [DiagMatrix](#).
- class [ExpandedMultiVectorMatrix](#)
Class for Matrices with few rows that consists of Vectors, together with a premultiplied Expansion matrix.
- class [ExpandedMultiVectorMatrixSpace](#)
This is the matrix space for [ExpandedMultiVectorMatrix](#).
- class [ExpansionMatrix](#)
Class for expansion/projection matrices.
- class [ExpansionMatrixSpace](#)
This is the matrix space for [ExpansionMatrix](#).
- class [IdentityMatrix](#)

- Class for Matrices which are multiples of the identity matrix.*

 - class [IdentityMatrixSpace](#)

This is the matrix space for [IdentityMatrix](#).
 - class [LowRankUpdateSymMatrix](#)

Class for symmetric matrices, represented as low-rank updates.
 - class [LowRankUpdateSymMatrixSpace](#)

This is the matrix space for [LowRankUpdateSymMatrix](#).
 - class [Matrix](#)

[Matrix](#) Base Class.
 - class [MatrixSpace](#)

[MatrixSpace](#) base class, corresponding to the [Matrix](#) base class.
 - class [MultiVectorMatrix](#)

Class for Matrices with few columns that consists of Vectors.
 - class [MultiVectorMatrixSpace](#)

This is the matrix space for [MultiVectorMatrix](#).
 - class [ScaledMatrix](#)

Class for a [Matrix](#) in conjunction with its scaling factors for row and column scaling.
 - class [ScaledMatrixSpace](#)

This is the matrix space for [ScaledMatrix](#).
 - class [SumMatrix](#)

Class for Matrices which are sum of matrices.
 - class [SumMatrixSpace](#)

Class for matrix space for [SumMatrix](#).
 - class [SumSymMatrix](#)

Class for Matrices which are sum of symmetric matrices.
 - class [SumSymMatrixSpace](#)

Class for matrix space for [SumSymMatrix](#).
 - class [SymMatrix](#)

This is the base class for all derived symmetric matrix types.
 - class [SymMatrixSpace](#)

[SymMatrixSpace](#) base class, corresponding to the [SymMatrix](#) base class.
 - class [SymScaledMatrix](#)

Class for a [Matrix](#) in conjunction with its scaling factors for row and column scaling.
 - class [SymScaledMatrixSpace](#)

This is the matrix space for [SymScaledMatrix](#).
 - class [TransposeMatrix](#)

Class for Matrices which are the transpose of another matrix.
 - class [TransposeMatrixSpace](#)

This is the matrix space for [TransposeMatrix](#).
 - class [Vector](#)

[Vector](#) Base Class.
 - class [VectorSpace](#)

[VectorSpace](#) base class, corresponding to the [Vector](#) base class.
 - class [ZeroMatrix](#)

Class for Matrices with only zero entries.
 - class [ZeroMatrixSpace](#)

Class for matrix space for [ZeroMatrix](#).

- class [ZeroSymMatrix](#)
Class for Symmetric Matrices with only zero entries.
- class [ZeroSymMatrixSpace](#)
Class for matrix space for [ZeroSymMatrix](#).
- class [GenTMatrix](#)
Class for general matrices stored in triplet format.
- class [GenTMatrixSpace](#)
This is the matrix space for a [GenTMatrix](#) with fixed sparsity structure.
- class [SymTMatrix](#)
Class for symmetric matrices stored in triplet format.
- class [SymTMatrixSpace](#)
This is the matrix space for a [SymTMatrix](#) with fixed sparsity structure.
- class [TripletHelper](#)

Typedefs

- typedef double [Number](#)
Type of all numbers.
- typedef int [Index](#)
Type of all indices of vectors, matrices etc.
- typedef int [Int](#)
Type of default integer.
- typedef struct
[Ipopt::PiecewisePenEntry](#) [PiecewisePenEntry](#)
struct for one Piecewise Penalty entry.
- typedef std::map< std::string,
std::vector< std::string > > [StringMetaDataMapType](#)
typedefs for the map variables that define meta data for the [DenseVectorSpace](#)
- typedef std::map< std::string,
std::vector< [Index](#) > > [IntegerMetaDataMapType](#)
- typedef std::map< std::string,
std::vector< [Number](#) > > [NumericMetaDataMapType](#)

Enumerations

- enum [ENormType](#) { [NORM_1](#) =0, [NORM_2](#), [NORM_MAX](#) }
Norm types.
- enum [HessianApproximationType](#) { [EXACT](#) =0, [LIMITED_MEMORY](#) }
enumeration for the Hessian information type.
- enum [HessianApproximationSpace](#) { [NONLINEAR_VARS](#) =0, [ALL_VARS](#) }
enumeration for the Hessian approximation space.
- enum [ESymSolverStatus](#) {
[SYMSOLVER_SUCCESS](#), [SYMSOLVER_SINGULAR](#), [SYMSOLVER_WRONG_INERTIA](#), [SYMSOLVER_CALL_AGAIN](#),
[SYMSOLVER_FATAL_ERROR](#) }
Enum to report outcome of a linear solve.
- enum [RegisteredOptionType](#) { [OT_Number](#), [OT_Integer](#), [OT_String](#), [OT_Unknown](#) }

- enum [ApplicationReturnStatus](#) {
[Solve_Succeeded](#) =0, [Solved_To_Acceptable_Level](#) =1, [Infeasible_Problem_Detected](#) =2, [Search_Direction_Becomes_Too_Small](#) =3,
[Diverging_Iterates](#) =4, [User_Requested_Stop](#) =5, [Feasible_Point_Found](#) =6, [Maximum_Iterations_Exceeded](#) =-1,
[Restoration_Failed](#) =-2, [Error_In_Step_Computation](#) =-3, [Maximum_CpuTime_Exceeded](#) =-4, [Not_Enough_Degrees_Of_Freedom](#) =-10,
[Invalid_Problem_Definition](#) =-11, [Invalid_Option](#) =-12, [Invalid_Number_Detected](#) =-13, [Unrecoverable_Exception](#) =-100,
[NonIpopt_Exception_Thrown](#) =-101, [Insufficient_Memory](#) =-102, [Internal_Error](#) =-199 }
Return codes for the Optimize call for an application.
- enum [AlgorithmMode](#) { [RegularMode](#) =0, [RestorationPhaseMode](#) =1 }
enum to indicate the mode in which the algorithm is

Journalist Enumerations.

- enum [EJournalLevel](#) {
[J_INSUPPRESSIBLE](#) =-1, [J_NONE](#) =0, [J_ERROR](#), [J_STRONGWARNING](#),
[J_SUMMARY](#), [J_WARNING](#), [J_ITERSUMMARY](#), [J_DETAILED](#),
[J_MOREDETAILED](#), [J_VECTOR](#), [J_MOREVECTOR](#), [J_MATRIX](#),
[J_MOREMATRIX](#), [J_ALL](#), [J_LAST_LEVEL](#) }
Print Level Enum.
- enum [EJournalCategory](#) {
[J_DBG](#) =0, [J_STATISTICS](#), [J_MAIN](#), [J_INITIALIZATION](#),
[J_BARRIER_UPDATE](#), [J_SOLVE_PD_SYSTEM](#), [J_FRAC_TO_BOUND](#), [J_LINEAR_ALGEBRA](#),
[J_LINE_SEARCH](#), [J_HESSIAN_APPROXIMATION](#), [J_SOLUTION](#), [J_DOCUMENTATION](#),
[J_NLP](#), [J_TIMING_STATISTICS](#), [J_USER_APPLICATION](#), [J_USER1](#),
[J_USER2](#), [J_USER3](#), [J_USER4](#), [J_USER5](#),
[J_USER6](#), [J_USER7](#), [J_USER8](#), [J_USER9](#),
[J_USER10](#), [J_USER11](#), [J_USER12](#), [J_USER13](#),
[J_USER14](#), [J_USER15](#), [J_USER16](#), [J_USER17](#),
[J_LAST_CATEGORY](#) }
Category Selection Enum.

Enumerations

- enum [SolverReturn](#) {
[SUCCESS](#), [MAXITER_EXCEEDED](#), [CPUTIME_EXCEEDED](#), [STOP_AT_TINY_STEP](#),
[STOP_AT_ACCEPTABLE_POINT](#), [LOCAL_INFEASIBILITY](#), [USER_REQUESTED_STOP](#), [FEASIBLE_POINT_FOUND](#),
[DIVERGING_ITERATES](#), [RESTORATION_FAILURE](#), [ERROR_IN_STEP_COMPUTATION](#), [INVALID_NUMBER_DETECTED](#),
[TOO_FEW_DEGREES_OF_FREEDOM](#), [INVALID_OPTION](#), [OUT_OF_MEMORY](#), [INTERNAL_ERROR](#),
[UNASSIGNED](#) }
enum for the return from the optimize algorithm (obviously we need to add more)

Functions

- void [AddInexactDefaultOptions](#) ([OptionsList](#) &options_list)
Function for setting options whos default is different for the inexact algorithm compared to the defaults for the regular [Ipopt](#) algorithm.
- void [RegisterOptions_Inexact](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)
- void [RegisterOptions_Algorithm](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)

- [DECLARE_STD_EXCEPTION](#) (FATAL_ERROR_IN_LINEAR_SOLVER)
- void [RegisterOptions_LinearSolvers](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)
- [DECLARE_STD_EXCEPTION](#) (ERROR_IN_LINEAR_SCALING_METHOD)
- [DECLARE_STD_EXCEPTION](#) (OPTION_INVALID)
 - Exception that can be used to indicate errors with options.*
- template<class U1 , class U2 >
 - bool [ComparePointers](#) (const U1 *lhs, const U2 *rhs)
- template<class T >
 - void [swap](#) ([SmartPtr](#)< T > &a, [SmartPtr](#)< T > &b)
- template<class T >
 - bool [operator](#)< (const [SmartPtr](#)< T > &lhs, const [SmartPtr](#)< T > &rhs)
- template<class T >
 - bool [operator](#)> (const [SmartPtr](#)< T > &lhs, const [SmartPtr](#)< T > &rhs)
- template<class T >
 - bool [operator](#)<= (const [SmartPtr](#)< T > &lhs, const [SmartPtr](#)< T > &rhs)
- template<class T >
 - bool [operator](#)>= (const [SmartPtr](#)< T > &lhs, const [SmartPtr](#)< T > &rhs)
- [TaggedObject::Tag operator+](#) (const [TaggedObject::Tag](#) &tag1, const [TaggedObject::Tag](#) &tag2)
 - The addition of two tags - do not use.*
- [Index Max](#) ([Index](#) a, [Index](#) b)
- [Index Max](#) ([Index](#) a, [Index](#) b, [Index](#) c)
- [Index Max](#) ([Index](#) a, [Index](#) b, [Index](#) c, [Index](#) d)
- [Index Min](#) ([Index](#) a, [Index](#) b)
- [Index Min](#) ([Index](#) a, [Index](#) b, [Index](#) c)
- [Index Min](#) ([Index](#) a, [Index](#) b, [Index](#) c, [Index](#) d)
- [Number Max](#) ([Number](#) a, [Number](#) b)
- [Number Max](#) ([Number](#) a, [Number](#) b, [Number](#) c)
- [Number Max](#) ([Number](#) a, [Number](#) b, [Number](#) c, [Number](#) d)
- [Number Min](#) ([Number](#) a, [Number](#) b)
- [Number Min](#) ([Number](#) a, [Number](#) b, [Number](#) c)
- [Number Min](#) ([Number](#) a, [Number](#) b, [Number](#) c, [Number](#) d)
- bool [IsFiniteNumber](#) ([Number](#) val)
 - Function returning true iff the argument is a valid double number (not NaN or Inf).*
- [Number lpRandom01](#) ()
 - Function returning a random number between 0 and 1.*
- void [lpResetRandom01](#) ()
 - Function resetting the random number generator.*
- [Number CpuTime](#) ()
 - method determining CPU time*
- [Number SysTime](#) ()
 - method determining system time*
- [Number WallclockTime](#) ()
 - method determining wallclock time since first call*
- bool [Compare_le](#) ([Number](#) lhs, [Number](#) rhs, [Number](#) BasVal)
 - Method for comparing two numbers within machine precision.*
- int [Snprintf](#) (char *str, long size, const char *format,...)
 - Method for printing a formatted output to a string with given size.*
- void [RegisterOptions_CGPenalty](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)
- void [RegisterOptions_Interfaces](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)

- [DECLARE_STD_EXCEPTION](#) (IPOPT_APPLICATION_ERROR)
- [DECLARE_STD_EXCEPTION](#) (INVALID_STDINTERFACE_NLP)

Declare exception that is thrown when invalid NLP data is provided.
- [Number IpBlasDdot](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX, const [Number](#) *y, [Index](#) incY)

Wrapper for BLAS function DDOT.
- [Number IpBlasDnrm2](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX)

Wrapper for BLAS function DNRM2.
- [Number IpBlasDasum](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX)

Wrapper for BLAS function DASUM.
- [Index IpBlasIdamax](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX)

Wrapper for BLAS function IDAMAX.
- void [IpBlasDcopy](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX, [Number](#) *y, [Index](#) incY)

Wrapper for BLAS subroutine DCOPY.
- void [IpBlasDaxpy](#) ([Index](#) size, [Number](#) alpha, const [Number](#) *x, [Index](#) incX, [Number](#) *y, [Index](#) incY)

Wrapper for BLAS subroutine DAXPY.
- void [IpBlasDscal](#) ([Index](#) size, [Number](#) alpha, [Number](#) *x, [Index](#) incX)

Wrapper for BLAS subroutine DSCAL.
- void [IpBlasDgemv](#) (bool trans, [Index](#) nRows, [Index](#) nCols, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, const [Number](#) *x, [Index](#) incX, [Number](#) beta, [Number](#) *y, [Index](#) incY)

Wrapper for BLAS subroutine DGEMV.
- void [IpBlasDsymv](#) ([Index](#) n, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, const [Number](#) *x, [Index](#) incX, [Number](#) beta, [Number](#) *y, [Index](#) incY)

Wrapper for BLAS subroutine DSYMV.
- void [IpBlasDgemm](#) (bool transa, bool transb, [Index](#) m, [Index](#) n, [Index](#) k, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, const [Number](#) *B, [Index](#) ldB, [Number](#) beta, [Number](#) *C, [Index](#) ldC)

Wrapper for BLAS subroutine DGEMM.
- void [IpBlasDsyrk](#) (bool trans, [Index](#) ndim, [Index](#) nrank, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, [Number](#) beta, [Number](#) *C, [Index](#) ldC)

Wrapper for BLAS subroutine DSYRK.
- void [IpBlasDtrsm](#) (bool trans, [Index](#) ndim, [Index](#) nrhs, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, [Number](#) *B, [Index](#) ldB)

Wrapper for BLAS subroutine DTRSM.
- [DECLARE_STD_EXCEPTION](#) (LAPACK_NOT_INCLUDED)
- void [IpLapackDpotrs](#) ([Index](#) ndim, [Index](#) nrhs, const [Number](#) *a, [Index](#) lda, [Number](#) *b, [Index](#) ldb)

Wrapper for LAPACK subroutine DPOTRS.
- void [IpLapackDpotrf](#) ([Index](#) ndim, [Number](#) *a, [Index](#) lda, [Index](#) &info)

Wrapper for LAPACK subroutine DPOTRF.
- void [IpLapackDsyev](#) (bool compute_eigenvalues, [Index](#) ndim, [Number](#) *a, [Index](#) lda, [Number](#) *w, [Index](#) &info)

Wrapper for LAPACK subroutine DSYEV.
- void [IpLapackDgetrf](#) ([Index](#) ndim, [Number](#) *a, [Index](#) *pivot, [Index](#) lda, [Index](#) &info)

Wrapper for LAPACK subroutine DGETRF.
- void [IpLapackDgetrs](#) ([Index](#) ndim, [Index](#) nrhs, const [Number](#) *a, [Index](#) lda, [Index](#) *ipiv, [Number](#) *b, [Index](#) ldb)

Wrapper for LAPACK subroutine DGETRS.
- [DECLARE_STD_EXCEPTION](#) (UNIMPLEMENTED_LINALG_METHOD_CALLED)

Exception that can be used to flag unimplemented linear algebra methods.
- [DECLARE_STD_EXCEPTION](#) (UNKNOWN_MATRIX_TYPE)
- [DECLARE_STD_EXCEPTION](#) (UNKNOWN_VECTOR_TYPE)

Exceptions

- `DECLARE_STD_EXCEPTION` (STEP_COMPUTATION_FAILED)
- `DECLARE_STD_EXCEPTION` (RESTORATION_CONVERGED_TO_FEASIBLE_POINT)
Exception RESTORATION_FAILED for all trouble with the restoration phase.
- `DECLARE_STD_EXCEPTION` (RESTORATION_FAILED)
- `DECLARE_STD_EXCEPTION` (RESTORATION_MAXITER_EXCEEDED)
- `DECLARE_STD_EXCEPTION` (RESTORATION_CPUTIME_EXCEEDED)
- `DECLARE_STD_EXCEPTION` (RESTORATION_USER_STOP)
- `DECLARE_STD_EXCEPTION` (METADATA_ERROR)

SmartPtr friend function declarations.

- `template<class U >`
`U * GetRawPtr (const SmartPtr< U > &smart_ptr)`
- `template<class U >`
`SmartPtr< const U > ConstPtr (const SmartPtr< U > &smart_ptr)`
- `template<class U >`
`bool IsNull (const SmartPtr< U > &smart_ptr)`
- `template<class U >`
`bool IsValid (const SmartPtr< U > &smart_ptr)`
- `template<class U1 , class U2 >`
`bool operator== (const SmartPtr< U1 > &lhs, const SmartPtr< U2 > &rhs)`
- `template<class U1 , class U2 >`
`bool operator== (const SmartPtr< U1 > &lhs, U2 *raw_rhs)`
- `template<class U1 , class U2 >`
`bool operator== (U1 *lhs, const SmartPtr< U2 > &raw_rhs)`
- `template<class U1 , class U2 >`
`bool operator!= (const SmartPtr< U1 > &lhs, const SmartPtr< U2 > &rhs)`
- `template<class U1 , class U2 >`
`bool operator!= (const SmartPtr< U1 > &lhs, U2 *raw_rhs)`
- `template<class U1 , class U2 >`
`bool operator!= (U1 *lhs, const SmartPtr< U2 > &raw_rhs)`

Some exceptions used in multiple places

- `DECLARE_STD_EXCEPTION` (LOCALLY_INFEASIBLE)
- `DECLARE_STD_EXCEPTION` (TOO_FEW_DOF)
- `DECLARE_STD_EXCEPTION` (TINY_STEP_DETECTED)
- `DECLARE_STD_EXCEPTION` (ACCEPTABLE_POINT_REACHED)
- `DECLARE_STD_EXCEPTION` (FEASIBILITY_PROBLEM_SOLVED)
- `DECLARE_STD_EXCEPTION` (INVALID_WARMSTART)
- `DECLARE_STD_EXCEPTION` (INTERNAL_ABORT)
- `DECLARE_STD_EXCEPTION` (NO_FREE_VARIABLES_BUT_FEASIBLE)
- `DECLARE_STD_EXCEPTION` (NO_FREE_VARIABLES_AND_INFEASIBLE)
- `DECLARE_STD_EXCEPTION` (FAILED_INITIALIZATION)

Exception FAILED_INITIALIZATION for problem during initialization of a strategy object (or other problems).

5.1.1 Typedef Documentation

5.1.1.1 typedef double Ipopt::Number

Type of all numbers.

Definition at line 17 of file IpTypes.hpp.

5.1.1.2 typedef int Ipopt::Index

Type of all indices of vectors, matrices etc.

Definition at line 19 of file IpTypes.hpp.

5.1.1.3 typedef int Ipopt::Int

Type of default integer.

Definition at line 21 of file IpTypes.hpp.

5.1.1.4 typedef struct Ipopt::PiecewisePenEntry Ipopt::PiecewisePenEntry

struct for one Piecewise Penalty entry.

5.1.1.5 typedef std::map<std::string, std::vector<std::string> > Ipopt::StringMetaDataMapType

typedefs for the map variables that define meta data for the [DenseVectorSpace](#)

Definition at line 279 of file IpDenseVector.hpp.

5.1.1.6 typedef std::map<std::string, std::vector<Index> > Ipopt::IntegerMetaDataMapType

Definition at line 280 of file IpDenseVector.hpp.

5.1.1.7 typedef std::map<std::string, std::vector<Number> > Ipopt::NumericMetaDataMapType

Definition at line 281 of file IpDenseVector.hpp.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum Ipopt::ENormType

Norm types.

Enumerator

NORM_1

NORM_2

NORM_MAX

Definition at line 29 of file IpIpoptCalculatedQuantities.hpp.

5.1.2.2 enum Ipopt::HessianApproximationType

enumeration for the Hessian information type.

Enumerator

EXACT

LIMITED_MEMORY

Definition at line 20 of file IpOrigIpoptNLP.hpp.

5.1.2.3 enum Ipopt::HessianApproximationSpace

enumeration for the Hessian approximation space.

Enumerator

NONLINEAR_VARS

ALL_VARS

Definition at line 26 of file IpOrigIpoptNLP.hpp.

5.1.2.4 enum Ipopt::ESymSolverStatus

Enum to report outcome of a linear solve.

Enumerator

SYMSOLVER_SUCCESS Successful solve.

SYMSOLVER_SINGULAR [Matrix](#) seems to be singular; solve was aborted.

SYMSOLVER_WRONG_INERTIA The number of negative eigenvalues is not correct.

SYMSOLVER_CALL_AGAIN Call the solver interface again after the matrix values have been restored.

SYMSOLVER_FATAL_ERROR Unrecoverable error in linear solver occurred. The optimization will be aborted.

Definition at line 21 of file IpSymLinearSolver.hpp.

5.1.2.5 enum Ipopt::EJournalLevel

Print Level Enum.

Enumerator

J_INSUPPRESSIBLE

J_NONE

J_ERROR

J_STRONGWARNING

J_SUMMARY

J_WARNING

J_ITERSUMMARY

J_DETAILED

J_MOREDETAILED

J_VECTOR

J_MOREVECTOR

J_MATRIX

J_MOREMATRIX

J_ALL

J_LAST_LEVEL

Definition at line 51 of file IpJournalist.hpp.

5.1.2.6 enum Ipopt::EJournalCategory

Category Selection Enum.

Enumerator

J_DBG

J_STATISTICS

J_MAIN

J_INITIALIZATION

J_BARRIER_UPDATE

J_SOLVE_PD_SYSTEM

J_FRAC_TO_BOUND

J_LINEAR_ALGEBRA

J_LINE_SEARCH

J_HESSIAN_APPROXIMATION

J_SOLUTION

J_DOCUMENTATION

J_NLP

J_TIMING_STATISTICS

J_USER_APPLICATION This can be used by the user's application.

J_USER1 This can be used by the user's application.

J_USER2 This can be used by the user's application.

J_USER3 This can be used by the user's application.

J_USER4 This can be used by the user's application.

J_USER5 This can be used by the user's application.

J_USER6 This can be used by the user's application.

J_USER7 This can be used by the user's application.

J_USER8 This can be used by the user's application.

J_USER9 This can be used by the user's application.

J_USER10 This can be used by the user's application.

J_USER11 This can be used by the user's application.

J_USER12 This can be used by the user's application.

J_USER13 This can be used by the user's application.

J_USER14 This can be used by the user's application.

J_USER15 This can be used by the user's application.

J_USER16 This can be used by the user's application.

J_USER17 This can be used by the user's application.

J_LAST_CATEGORY

Definition at line 70 of file IpJournalist.hpp.

5.1.2.7 enum `Ipopt::RegisteredOptionType`

Enumerator

OT_Number
OT_Integer
OT_String
OT_Unknown

Definition at line 22 of file `IpRegOptions.hpp`.

5.1.2.8 enum `Ipopt::SolverReturn`

enum for the return from the optimize algorithm (obviously we need to add more)

Enumerator

SUCCESS
MAXITER_EXCEEDED
CPUTIME_EXCEEDED
STOP_AT_TINY_STEP
STOP_AT_ACCEPTABLE_POINT
LOCAL_INFEASIBILITY
USER_REQUESTED_STOP
FEASIBLE_POINT_FOUND
DIVERGING_ITERATES
RESTORATION_FAILURE
ERROR_IN_STEP_COMPUTATION
INVALID_NUMBER_DETECTED
TOO_FEW_DEGREES_OF_FREEDOM
INVALID_OPTION
OUT_OF_MEMORY
INTERNAL_ERROR
UNASSIGNED

Definition at line 22 of file `IpAlgTypes.hpp`.

5.1.2.9 enum `Ipopt::ApplicationReturnStatus`

Return codes for the Optimize call for an application.

Enumerator

Solve_Succeeded
Solved_To_Acceptable_Level
Infeasible_Problem_Detected
Search_Direction_Becomes_Too_Small
Diverging_Iterates
User_Requested_Stop

Feasible_Point_Found
Maximum_Iterations_Exceeded
Restoration_Failed
Error_In_Step_Computation
Maximum_CpuTime_Exceeded
Not_Enough_Degrees_Of_Freedom
Invalid_Problem_Definition
Invalid_Option
Invalid_Number_Detected
Unrecoverable_Exception
NonIpopt_Exception_Thrown
Insufficient_Memory
Internal_Error

Definition at line 17 of file IpReturnCodes.hpp.

5.1.2.10 enum Ipopt::AlgorithmMode

enum to indicate the mode in which the algorithm is

Enumerator

RegularMode
RestorationPhaseMode

Definition at line 43 of file IpReturnCodes.hpp.

5.1.3 Function Documentation

5.1.3.1 void Ipopt::AddInexactDefaultOptions (OptionsList & options_list)

Function for setting options whos default is different for the inexact algorithm compared to the defaults for the regular [Ipopt](#) algorithm.

The options_list is augmented by the different default values, but only if the corresponding option has not yet been set.

5.1.3.2 void Ipopt::RegisterOptions_Inexact (const SmartPtr< RegisteredOptions > & roptions)

5.1.3.3 void Ipopt::RegisterOptions_Algorithm (const SmartPtr< RegisteredOptions > & roptions)

5.1.3.4 Ipopt::DECLARE_STD_EXCEPTION (FATAL_ERROR_IN_LINEAR_SOLVER)

5.1.3.5 Ipopt::DECLARE_STD_EXCEPTION (STEP_COMPUTATION_FAILED)

5.1.3.6 Ipopt::DECLARE_STD_EXCEPTION (RESTORATION_CONVERGED_TO_FEASIBLE_POINT)

Exception RESTORATION_FAILED for all trouble with the restoration phase.

5.1.3.7 `Ipopt::DECLARE_STD_EXCEPTION (RESTORATION_FAILED)`

5.1.3.8 `Ipopt::DECLARE_STD_EXCEPTION (RESTORATION_MAXITER_EXCEEDED)`

5.1.3.9 `Ipopt::DECLARE_STD_EXCEPTION (RESTORATION_CPUTIME_EXCEEDED)`

5.1.3.10 `Ipopt::DECLARE_STD_EXCEPTION (RESTORATION_USER_STOP)`

5.1.3.11 `void Ipopt::RegisterOptions_LinearSolvers (const SmartPtr< RegisteredOptions > & options)`

5.1.3.12 `Ipopt::DECLARE_STD_EXCEPTION (ERROR_IN_LINEAR_SCALING_METHOD)`

5.1.3.13 `Ipopt::DECLARE_STD_EXCEPTION (OPTION_INVALID)`

Exception that can be used to indicate errors with options.

5.1.3.14 `template<class U> U * Ipopt::GetRawPtr (const SmartPtr< U > & smart_ptr)`

Use to get the value of the raw ptr (i.e. to pass to other methods/functions, etc.) Note: This method does NOT copy, therefore, modifications using this value modify the underlying object contained by the [SmartPtr](#), NEVER delete this returned value.

Definition at line 560 of file `IpSmartPtr.hpp`.

5.1.3.15 `template<class U> SmartPtr< const U > Ipopt::ConstPtr (const SmartPtr< U > & smart_ptr)`

Definition at line 572 of file `IpSmartPtr.hpp`.

5.1.3.16 `template<class U> bool Ipopt::IsNull (const SmartPtr< U > & smart_ptr)`

Use this to check if the [SmartPtr](#) IsNull. This is preferred to `if(GetRawPtr(sp) == NULL)`

Definition at line 585 of file `IpSmartPtr.hpp`.

5.1.3.17 `template<class U> bool Ipopt::IsValid (const SmartPtr< U > & smart_ptr)`

Use this to check if the [SmartPtr](#) is not null This is preferred to `if(GetRawPtr(sp) != NULL)`

Definition at line 579 of file `IpSmartPtr.hpp`.

5.1.3.18 `template<class U1, class U2> bool Ipopt::operator== (const SmartPtr< U1 > & lhs, const SmartPtr< U2 > & rhs)`

Definition at line 617 of file `IpSmartPtr.hpp`.

5.1.3.19 `template<class U1, class U2> bool Ipopt::operator== (const SmartPtr< U1 > & lhs, U2 * raw_rhs)`

Definition at line 631 of file `IpSmartPtr.hpp`.

5.1.3.20 `template<class U1, class U2> bool Ipopt::operator== (U1 * lhs, const SmartPtr< U2 > & raw_rhs)`

Definition at line 644 of file `IpSmartPtr.hpp`.

5.1.3.21 `template<class U1, class U2> bool Ipopt::operator!= (const SmartPtr< U1 > & lhs, const SmartPtr< U2 > & rhs)`

Definition at line 657 of file `IpSmartPtr.hpp`.

5.1.3.22 `template<class U1, class U2> bool Ipopt::operator!= (const SmartPtr< U1 > & lhs, U2 * raw_rhs)`

Definition at line 670 of file `IpSmartPtr.hpp`.

5.1.3.23 `template<class U1 , class U2 > bool Ipopt::operator!= (U1 * lhs, const SmartPtr< U2 > & raw_rhs)`

Definition at line 683 of file IpSmartPtr.hpp.

5.1.3.24 `template<class U1 , class U2 > bool Ipopt::ComparePointers (const U1 * lhs, const U2 * rhs)`

Definition at line 598 of file IpSmartPtr.hpp.

5.1.3.25 `template<class T > void Ipopt::swap (SmartPtr< T > & a, SmartPtr< T > & b)`

Definition at line 696 of file IpSmartPtr.hpp.

5.1.3.26 `template<class T > bool Ipopt::operator< (const SmartPtr< T > & lhs, const SmartPtr< T > & rhs)`

Definition at line 708 of file IpSmartPtr.hpp.

5.1.3.27 `template<class T > bool Ipopt::operator> (const SmartPtr< T > & lhs, const SmartPtr< T > & rhs)`

Definition at line 714 of file IpSmartPtr.hpp.

5.1.3.28 `template<class T > bool Ipopt::operator<= (const SmartPtr< T > & lhs, const SmartPtr< T > & rhs)`

Definition at line 720 of file IpSmartPtr.hpp.

5.1.3.29 `template<class T > bool Ipopt::operator>= (const SmartPtr< T > & lhs, const SmartPtr< T > & rhs)`

Definition at line 726 of file IpSmartPtr.hpp.

5.1.3.30 `TaggedObject::Tag Ipopt::operator+ (const TaggedObject::Tag & tag1, const TaggedObject::Tag & tag2)`
[inline]

The addition of two tags - do not use.

Note

Do not use this operator, unless you really know what you are doing.

Definition at line 153 of file IpTaggedObject.hpp.

5.1.3.31 `Index Ipopt::Max (Index a, Index b)` [inline]

Definition at line 19 of file IpUtils.hpp.

5.1.3.32 `Index Ipopt::Max (Index a, Index b, Index c)` [inline]

Definition at line 24 of file IpUtils.hpp.

5.1.3.33 `Index Ipopt::Max (Index a, Index b, Index c, Index d)` [inline]

Definition at line 31 of file IpUtils.hpp.

5.1.3.34 `Index Ipopt::Min (Index a, Index b)` [inline]

Definition at line 38 of file IpUtils.hpp.

5.1.3.35 `Index Ipopt::Min (Index a, Index b, Index c)` [inline]

Definition at line 43 of file IpUtils.hpp.

5.1.3.36 `Index Ipopt::Min (Index a, Index b, Index c, Index d)` `[inline]`

Definition at line 50 of file IpUtils.hpp.

5.1.3.37 `Number Ipopt::Max (Number a, Number b)` `[inline]`

Definition at line 59 of file IpUtils.hpp.

5.1.3.38 `Number Ipopt::Max (Number a, Number b, Number c)` `[inline]`

Definition at line 64 of file IpUtils.hpp.

5.1.3.39 `Number Ipopt::Max (Number a, Number b, Number c, Number d)` `[inline]`

Definition at line 71 of file IpUtils.hpp.

5.1.3.40 `Number Ipopt::Min (Number a, Number b)` `[inline]`

Definition at line 78 of file IpUtils.hpp.

5.1.3.41 `Number Ipopt::Min (Number a, Number b, Number c)` `[inline]`

Definition at line 83 of file IpUtils.hpp.

5.1.3.42 `Number Ipopt::Min (Number a, Number b, Number c, Number d)` `[inline]`

Definition at line 90 of file IpUtils.hpp.

5.1.3.43 `bool Ipopt::IsFiniteNumber (Number val)`

Function returning true iff the argument is a valid double number (not NaN or Inf).

5.1.3.44 `Number Ipopt::IpRandom01 ()`

Function returning a random number between 0 and 1.

5.1.3.45 `void Ipopt::IpResetRandom01 ()`

Function resetting the random number generator.

5.1.3.46 `Number Ipopt::CpuTime ()`

method determining CPU time

5.1.3.47 `Number Ipopt::SysTime ()`

method determining system time

5.1.3.48 `Number Ipopt::WallclockTime ()`

method determining wallclock time since first call

5.1.3.49 `bool Ipopt::Compare_le (Number lhs, Number rhs, Number BasVal)`

Method for comparing two numbers within machine precision.

The return value is true if lhs is less or equal the rhs, relaxing this inequality by something a little larger than machine precision relative to the absolute value of BasVal.

5.1.3.50 `int Ipopt::Sprintf (char * str, long size, const char * format, ...)`

Method for printing a formatted output to a string with given size.

5.1.3.51 `void Ipopt::RegisterOptions_CGPenalty (const SmartPtr< RegisteredOptions > & roptions)`

5.1.3.52 `Ipopt::DECLARE_STD_EXCEPTION (LOCALLY_INFEASIBLE)`

5.1.3.53 `Ipopt::DECLARE_STD_EXCEPTION (TOO_FEW_DOF)`

5.1.3.54 `Ipopt::DECLARE_STD_EXCEPTION (TINY_STEP_DETECTED)`

5.1.3.55 `Ipopt::DECLARE_STD_EXCEPTION (ACCEPTABLE_POINT_REACHED)`

5.1.3.56 `Ipopt::DECLARE_STD_EXCEPTION (FEASIBILITY_PROBLEM_SOLVED)`

5.1.3.57 `Ipopt::DECLARE_STD_EXCEPTION (INVALID_WARMSTART)`

5.1.3.58 `Ipopt::DECLARE_STD_EXCEPTION (INTERNAL_ABORT)`

5.1.3.59 `Ipopt::DECLARE_STD_EXCEPTION (NO_FREE_VARIABLES_BUT_FEASIBLE)`

5.1.3.60 `Ipopt::DECLARE_STD_EXCEPTION (NO_FREE_VARIABLES_AND_INFEASIBLE)`

5.1.3.61 `Ipopt::DECLARE_STD_EXCEPTION (FAILED_INITIALIZATION)`

Exception FAILED_INITIALIZATION for problem during initialization of a strategy object (or other problems).

This is thrown by a strategy object, if a problem arises during initialization, such as a value out of a feasible range.

5.1.3.62 `void Ipopt::RegisterOptions_Interfaces (const SmartPtr< RegisteredOptions > & roptions)`

5.1.3.63 `Ipopt::DECLARE_STD_EXCEPTION (IPOPT_APPLICATION_ERROR)`

5.1.3.64 `Ipopt::DECLARE_STD_EXCEPTION (INVALID_STDINTERFACE_NLP)`

Declare exception that is thrown when invalid [NLP](#) data is provided.

5.1.3.65 `Number Ipopt::IpBlasDdot (Index size, const Number * x, Index incX, const Number * y, Index incY)`

Wrapper for BLAS function DDOT.

Compute dot product of vector x and vector y

5.1.3.66 `Number Ipopt::IpBlasDnrm2 (Index size, const Number * x, Index incX)`

Wrapper for BLAS function DNRM2.

Compute 2-norm of vector x

5.1.3.67 `Number Ipopt::IpBlasDasum (Index size, const Number * x, Index incX)`

Wrapper for BLAS function DASUM.

Compute 1-norm of vector x

5.1.3.68 `Index Ipopt::IpBlasIdamax (Index size, const Number * x, Index incX)`

Wrapper for BLAS function IDAMAX.

Compute index for largest absolute element of vector x

5.1.3.69 `void lpopt::lpBlasDcopy (Index size, const Number * x, Index incX, Number * y, Index incY)`

Wrapper for BLAS subroutine DCOPY.

Copying vector x into vector y

5.1.3.70 `void lpopt::lpBlasDaxpy (Index size, Number alpha, const Number * x, Index incX, Number * y, Index incY)`

Wrapper for BLAS subroutine DAXPY.

Adding the alpha multiple of vector x to vector y

5.1.3.71 `void lpopt::lpBlasDscal (Index size, Number alpha, Number * x, Index incX)`

Wrapper for BLAS subroutine DSCAL.

Scaling vector x by scalar alpha

5.1.3.72 `void lpopt::lpBlasDgemv (bool trans, Index nRows, Index nCols, Number alpha, const Number * A, Index ldA, const Number * x, Index incX, Number beta, Number * y, Index incY)`

Wrapper for BLAS subroutine DGEMV.

Multiplying a matrix with a vector.

5.1.3.73 `void lpopt::lpBlasDsymv (Index n, Number alpha, const Number * A, Index ldA, const Number * x, Index incX, Number beta, Number * y, Index incY)`

Wrapper for BLAS subroutine DSYMV.

Multiplying a symmetric matrix with a vector.

5.1.3.74 `void lpopt::lpBlasDgemm (bool transa, bool transb, Index m, Index n, Index k, Number alpha, const Number * A, Index ldA, const Number * B, Index ldB, Number beta, Number * C, Index ldC)`

Wrapper for BLAS subroutine DGEMM.

Multiplying two matrices

5.1.3.75 `void lpopt::lpBlasDsyrk (bool trans, Index ndim, Index nrank, Number alpha, const Number * A, Index ldA, Number beta, Number * C, Index ldC)`

Wrapper for BLAS subroutine DSYRK.

Adding a high-rank update to a matrix

5.1.3.76 `void lpopt::lpBlasDtrsm (bool trans, Index ndim, Index nrhs, Number alpha, const Number * A, Index ldA, Number * B, Index ldB)`

Wrapper for BLAS subroutine DTRSM.

Backsolve for a lower triangular matrix.

5.1.3.77 `lpopt::DECLARE_STD_EXCEPTION (METADATA_ERROR)`

5.1.3.78 `lpopt::DECLARE_STD_EXCEPTION (LAPACK_NOT_INCLUDED)`

5.1.3.79 `void Ipopt::IpLapackDpotrs (Index ndim, Index nrhs, const Number * a, Index lda, Number * b, Index ldb)`

Wrapper for LAPACK subroutine DPOTRS.

Solving a linear system given a Cholesky factorization. We assume that the Cholesky factor is lower triangular.

5.1.3.80 `void Ipopt::IpLapackDpotrf (Index ndim, Number * a, Index lda, Index & info)`

Wrapper for LAPACK subroutine DPOTRF.

Compute Cholesky factorization (lower triangular factor). *info* is the return value from the LAPACK routine.

5.1.3.81 `void Ipopt::IpLapackDsyev (bool compute_eigenvectors, Index ndim, Number * a, Index lda, Number * w, Index & info)`

Wrapper for LAPACK subroutine DSYEV.

Compute the Eigenvalue decomposition for a given matrix. If *compute_eigenvectors* is true, *a* will contain the eigenvectors in its columns on return.

5.1.3.82 `void Ipopt::IpLapackDgetrf (Index ndim, Number * a, Index * pivot, Index lda, Index & info)`

Wrapper for LAPACK subroutine DGETRF.

Compute LU factorization. *info* is the return value from the LAPACK routine.

5.1.3.83 `void Ipopt::IpLapackDgetrs (Index ndim, Index nrhs, const Number * a, Index lda, Index * ipiv, Number * b, Index ldb)`

Wrapper for LAPACK subroutine DGETRS.

Solving a linear system given a LU factorization.

5.1.3.84 `Ipopt::DECLARE_STD_EXCEPTION (UNIMPLEMENTED_LINALG_METHOD_CALLED)`

Exception that can be used to flag unimplemented linear algebra methods.

5.1.3.85 `Ipopt::DECLARE_STD_EXCEPTION (UNKNOWN_MATRIX_TYPE)`

5.1.3.86 `Ipopt::DECLARE_STD_EXCEPTION (UNKNOWN_VECTOR_TYPE)`

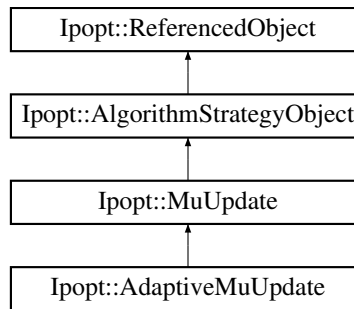
6 Class Documentation

6.1 Ipopt::AdaptiveMuUpdate Class Reference

Non-monotone mu update.

```
#include <IpAdaptiveMuUpdate.hpp>
```

Inheritance diagram for Ipopt::AdaptiveMuUpdate:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
Initialize method - overloaded from [AlgorithmStrategyObject](#).
- virtual bool [UpdateBarrierParameter](#) ()
Method for determining the barrier parameter for the next iteration.

Constructors/Destructors

- [AdaptiveMuUpdate](#) (const [SmartPtr](#)< [LineSearch](#) > &linesearch, const [SmartPtr](#)< [MuOracle](#) > &free_mu_oracle, const [SmartPtr](#)< [MuOracle](#) > &fix_mu_oracle=NULL)
Constructor.
- virtual [~AdaptiveMuUpdate](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [AdaptiveMuUpdate](#) ()
Default Constructor.
- [AdaptiveMuUpdate](#) (const [AdaptiveMuUpdate](#) &)
Copy Constructor.
- void [operator=](#) (const [AdaptiveMuUpdate](#) &)
Overloaded Equals Operator.

Private Attributes

- [Number init_dual_inf_](#)
Dual infeasibility at initial point.
- [Number init_primal_inf_](#)

Primal infeasibility at initial point.

- bool [no_bounds_](#)

Flag indicating whether the problem has any inequality constraints.

- bool [check_if_no_bounds_](#)

Flag indicating whether no_bounds_ has been initialized.

Strategy objects

- [SmartPtr](#)< [LineSearch](#) > [linesearch_](#)

Line search object of the [Ipopt](#) algorithm.

- [SmartPtr](#)< [MuOracle](#) > [free_mu_oracle_](#)

Pointer to strategy object that is to be used for computing a suggested value of the barrier parameter in the free mu mode.

- [SmartPtr](#)< [MuOracle](#) > [fix_mu_oracle_](#)

Pointer to strategy object that is to be used for computing a suggested value for the fixed mu mode.

Most recent accepted point in free mode, from which

fixed mode should be started.

- [SmartPtr](#)< const [IteratesVector](#) > [accepted_point_](#)

Algorithmic parameters

- enum [AdaptiveMuGlobalizationEnum](#) { [KKT_ERROR](#) =0, [FILTER_OBJ_CONSTR](#), [NEVER_MONOTONE_MODE](#) }

enumeration for adaptive globalization

- [Number](#) [mu_max_fact_](#)
- [Number](#) [mu_max_](#)
- [Number](#) [mu_min_](#)
- [Number](#) [mu_target_](#)
- bool [mu_min_default_](#)
- [Number](#) [tau_min_](#)
- [Number](#) [adaptive_mu_safeguard_factor_](#)
- [Number](#) [adaptive_mu_monotone_init_factor_](#)
- [Number](#) [barrier_tol_factor_](#)
- [Number](#) [mu_linear_decrease_factor_](#)
- [Number](#) [mu_superlinear_decrease_power_](#)
- [QualityFunctionMuOracle::NormEnum](#) [adaptive_mu_kkt_norm_](#)
- [QualityFunctionMuOracle::CentralityEnum](#) [adaptive_mu_kkt_centrality_](#)
- [QualityFunctionMuOracle::BalancingTermEnum](#) [adaptive_mu_kkt_balancing_term_](#)
- [AdaptiveMuGlobalizationEnum](#) [adaptive_mu_globalization_](#)

Flag indicating which globalization strategy should be used.

- [Number](#) [filter_max_margin_](#)

Maximal margin in filter.

- [Number](#) [filter_margin_fact_](#)

Factor for filter margin.

- [Number](#) [compl_inf_tol_](#)

Unscaled tolerance for complementarity.

Methods and data defining the outer globalization

strategy (might be a strategy object later).

- [Index num_refs_max_](#)
Maximal number of reference values (algorithmic parameter)
- `std::list< Number > refs_vals_`
Values of the currently stored reference values (norm of pd equations)
- [Number refs_red_fact_](#)
Factor requested to reduce the reference values.
- [Filter filter_](#)
Alternatively, we might also want to use a filter.
- `bool restore_accepted_iterate_`
Flag indicating whether the most recent accepted step should be restored, when switching to the fixed mode.
- `void InitializeFixedMuGlobalization ()`
- `bool CheckSufficientProgress ()`
Check whether the point in the "current" fields offers sufficient reduction in order to remain in or switch to the free mu mode.
- `void RememberCurrentPointAsAccepted ()`
Include the current point in internal memory to as accepted point.
- [Number NewFixedMu \(\)](#)
Compute the value of the fixed mu that should be used in a new fixed mu phase.
- [Number Compute_tau_monotone \(Number mu\)](#)
Compute value for the fraction-to-the-boundary parameter given mu in the monotone phase.
- [Number quality_function_pd_system \(\)](#)
Method for computing the norm of the primal dual system at the current point.
- [Number lower_mu_safeguard \(\)](#)
Method for computing a lower safeguard bound for the barrier parameter.
- [Number max_ref_val \(\)](#)
Computer the currently largest reference value.
- [Number min_ref_val \(\)](#)
Computer the currently smallest reference value.

Additional Inherited Members

6.1.1 Detailed Description

Non-monotone mu update.

Definition at line 23 of file IpAdaptiveMuUpdate.hpp.

6.1.2 Member Enumeration Documentation

6.1.2.1 `enum Ipopt::AdaptiveMuUpdate::AdaptiveMuGlobalizationEnum` `[private]`

enumeration for adaptive globalization

Enumerator

KKT_ERROR

FILTER_OBJ_CONSTR
NEVER_MONOTONE_MODE

Definition at line 89 of file IpAdaptiveMuUpdate.hpp.

6.1.3 Constructor & Destructor Documentation

6.1.3.1 **Ipopt::AdaptiveMuUpdate::AdaptiveMuUpdate (const SmartPtr< LineSearch > & *linesearch*, const SmartPtr< MuOracle > & *free_mu_oracle*, const SmartPtr< MuOracle > & *fix_mu_oracle* = NULL)**

Constructor.

6.1.3.2 **virtual Ipopt::AdaptiveMuUpdate::~AdaptiveMuUpdate () [virtual]**

Default destructor.

6.1.3.3 **Ipopt::AdaptiveMuUpdate::AdaptiveMuUpdate () [private]**

Default Constructor.

6.1.3.4 **Ipopt::AdaptiveMuUpdate::AdaptiveMuUpdate (const AdaptiveMuUpdate &) [private]**

Copy Constructor.

6.1.4 Member Function Documentation

6.1.4.1 **virtual bool Ipopt::AdaptiveMuUpdate::InitializeImpl (const OptionsList & *options*, const std::string & *prefix*) [virtual]**

Initialize method - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::MuUpdate](#).

6.1.4.2 **virtual bool Ipopt::AdaptiveMuUpdate::UpdateBarrierParameter () [virtual]**

Method for determining the barrier parameter for the next iteration.

When the optimality error for the current barrier parameter is less than a tolerance, the barrier parameter is reduced, and the Reset method of the [LineSearch](#) object linesearch is called.

Implements [Ipopt::MuUpdate](#).

6.1.4.3 **static void Ipopt::AdaptiveMuUpdate::RegisterOptions (SmartPtr< RegisteredOptions > *roptions*) [static]**

Methods for IpoptType.

6.1.4.4 **void Ipopt::AdaptiveMuUpdate::operator= (const AdaptiveMuUpdate &) [private]**

Overloaded Equals Operator.

6.1.4.5 **void Ipopt::AdaptiveMuUpdate::InitializeFixedMuGlobalization () [private]**

6.1.4.6 **bool Ipopt::AdaptiveMuUpdate::CheckSufficientProgress () [private]**

Check whether the point in the "current" fields offers sufficient reduction in order to remain in or switch to the free mu mode.

6.1.4.7 `void Ipopt::AdaptiveMuUpdate::RememberCurrentPointAsAccepted () [private]`

Include the current point in internal memory to as accepted point.

6.1.4.8 `Number Ipopt::AdaptiveMuUpdate::NewFixedMu () [private]`

Compute the value of the fixed mu that should be used in a new fixed mu phase.

This method is called at the beginning of a new fixed mu phase.

6.1.4.9 `Number Ipopt::AdaptiveMuUpdate::Compute_tau_monotone (Number mu) [private]`

Compute value for the fraction-to-the-boundary parameter given mu in the monotone phase.

6.1.4.10 `Number Ipopt::AdaptiveMuUpdate::quality_function_pd_system () [private]`

Method for computing the norm of the primal dual system at the current point.

For consistency, this is computed in the same way as the quality function is computed. This is the quantities used in the nonmonotone KKT reduction globalization.

6.1.4.11 `Number Ipopt::AdaptiveMuUpdate::lower_mu_safeguard () [private]`

Method for computing a lower safeguard bound for the barrier parameter.

For now, this is related to primal and dual infeasibility.

6.1.4.12 `Number Ipopt::AdaptiveMuUpdate::max_ref_val () [private]`

Computer the currently largest reference value.

6.1.4.13 `Number Ipopt::AdaptiveMuUpdate::min_ref_val () [private]`

Computer the currently smallest reference value.

6.1.5 Member Data Documentation

6.1.5.1 `Number Ipopt::AdaptiveMuUpdate::mu_max_fact_ [private]`

Definition at line 74 of file IpAdaptiveMuUpdate.hpp.

6.1.5.2 `Number Ipopt::AdaptiveMuUpdate::mu_max_ [private]`

Definition at line 75 of file IpAdaptiveMuUpdate.hpp.

6.1.5.3 `Number Ipopt::AdaptiveMuUpdate::mu_min_ [private]`

Definition at line 76 of file IpAdaptiveMuUpdate.hpp.

6.1.5.4 `Number Ipopt::AdaptiveMuUpdate::mu_target_ [private]`

Definition at line 77 of file IpAdaptiveMuUpdate.hpp.

6.1.5.5 `bool Ipopt::AdaptiveMuUpdate::mu_min_default_ [private]`

Definition at line 78 of file IpAdaptiveMuUpdate.hpp.

6.1.5.6 Number Ipopt::AdaptiveMuUpdate::tau_min_ [private]

Definition at line 79 of file IpAdaptiveMuUpdate.hpp.

6.1.5.7 Number Ipopt::AdaptiveMuUpdate::adaptive_mu_safeguard_factor_ [private]

Definition at line 80 of file IpAdaptiveMuUpdate.hpp.

6.1.5.8 Number Ipopt::AdaptiveMuUpdate::adaptive_mu_monotone_init_factor_ [private]

Definition at line 81 of file IpAdaptiveMuUpdate.hpp.

6.1.5.9 Number Ipopt::AdaptiveMuUpdate::barrier_tol_factor_ [private]

Definition at line 82 of file IpAdaptiveMuUpdate.hpp.

6.1.5.10 Number Ipopt::AdaptiveMuUpdate::mu_linear_decrease_factor_ [private]

Definition at line 83 of file IpAdaptiveMuUpdate.hpp.

6.1.5.11 Number Ipopt::AdaptiveMuUpdate::mu_superlinear_decrease_power_ [private]

Definition at line 84 of file IpAdaptiveMuUpdate.hpp.

6.1.5.12 QualityFunctionMuOracle::NormEnum Ipopt::AdaptiveMuUpdate::adaptive_mu_kkt_norm_ [private]

Definition at line 85 of file IpAdaptiveMuUpdate.hpp.

6.1.5.13 QualityFunctionMuOracle::CentralityEnum Ipopt::AdaptiveMuUpdate::adaptive_mu_kkt_centrality_ [private]

Definition at line 86 of file IpAdaptiveMuUpdate.hpp.

6.1.5.14 QualityFunctionMuOracle::BalancingTermEnum Ipopt::AdaptiveMuUpdate::adaptive_mu_kkt_balancing_term_ [private]

Definition at line 87 of file IpAdaptiveMuUpdate.hpp.

6.1.5.15 AdaptiveMuGlobalizationEnum Ipopt::AdaptiveMuUpdate::adaptive_mu_globalization_ [private]

Flag indicating which globalization strategy should be used.

Definition at line 96 of file IpAdaptiveMuUpdate.hpp.

6.1.5.16 Number Ipopt::AdaptiveMuUpdate::filter_max_margin_ [private]

Maximal margin in filter.

Definition at line 98 of file IpAdaptiveMuUpdate.hpp.

6.1.5.17 Number Ipopt::AdaptiveMuUpdate::filter_margin_fact_ [private]

Factor for filter margin.

Definition at line 100 of file IpAdaptiveMuUpdate.hpp.

6.1.5.18 Number Ipopt::AdaptiveMuUpdate::compl_inf_tol_ [private]

Unscaled tolerance for complementarity.

Definition at line 102 of file IpAdaptiveMuUpdate.hpp.

6.1.5.19 SmartPtr<LineSearch> Ipopt::AdaptiveMuUpdate::linesearch_ [private]

Line search object of the [Ipopt](#) algorithm.

Definition at line 108 of file IpAdaptiveMuUpdate.hpp.

6.1.5.20 SmartPtr<MuOracle> Ipopt::AdaptiveMuUpdate::free_mu_oracle_ [private]

Pointer to strategy object that is to be used for computing a suggested value of the barrier parameter in the free mu mode.

Definition at line 112 of file IpAdaptiveMuUpdate.hpp.

6.1.5.21 SmartPtr<MuOracle> Ipopt::AdaptiveMuUpdate::fix_mu_oracle_ [private]

Pointer to strategy object that is to be used for computing a suggested value for the fixed mu mode.

If NULL, the current average complementarity is used.

Definition at line 117 of file IpAdaptiveMuUpdate.hpp.

6.1.5.22 Number Ipopt::AdaptiveMuUpdate::init_dual_inf_ [private]

Dual infeasibility at initial point.

A negative value means that this quantity has not yet been initialized.

Definition at line 122 of file IpAdaptiveMuUpdate.hpp.

6.1.5.23 Number Ipopt::AdaptiveMuUpdate::init_primal_inf_ [private]

Primal infeasibility at initial point.

A negative value means that this quantity has not yet been initialized.

Definition at line 125 of file IpAdaptiveMuUpdate.hpp.

6.1.5.24 Index Ipopt::AdaptiveMuUpdate::num_refs_max_ [private]

Maximal number of reference values (algorithmic parameter)

Definition at line 165 of file IpAdaptiveMuUpdate.hpp.

6.1.5.25 std::list<Number> Ipopt::AdaptiveMuUpdate::refs_vals_ [private]

Values of the currently stored reference values (norm of pd equations)

Definition at line 168 of file IpAdaptiveMuUpdate.hpp.

6.1.5.26 Number Ipopt::AdaptiveMuUpdate::refs_red_fact_ [private]

Factor requested to reduce the reference values.

Definition at line 170 of file IpAdaptiveMuUpdate.hpp.

6.1.5.27 Filter Ipopt::AdaptiveMuUpdate::filter_ [private]

Alternatively, we might also want to use a filter.

Definition at line 173 of file IpAdaptiveMuUpdate.hpp.

6.1.5.28 `bool Ipopt::AdaptiveMuUpdate::restore_accepted_iterate_ [private]`

Flag indicating whether the most recent accepted step should be restored, when switching to the fixed mode.

Definition at line 176 of file `IpAdaptiveMuUpdate.hpp`.

6.1.5.29 `bool Ipopt::AdaptiveMuUpdate::no_bounds_ [private]`

Flag indicating whether the problem has any inequality constraints.

Definition at line 180 of file `IpAdaptiveMuUpdate.hpp`.

6.1.5.30 `bool Ipopt::AdaptiveMuUpdate::check_if_no_bounds_ [private]`

Flag indicating whether `no_bounds_` has been initialized.

Definition at line 182 of file `IpAdaptiveMuUpdate.hpp`.

6.1.5.31 `SmartPointer<const IteratesVector> Ipopt::AdaptiveMuUpdate::accepted_point_ [private]`

Definition at line 188 of file `IpAdaptiveMuUpdate.hpp`.

The documentation for this class was generated from the following file:

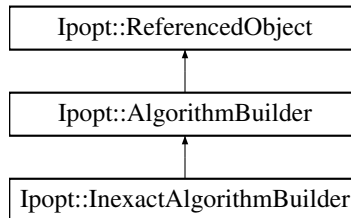
- [Algorithm/IpAdaptiveMuUpdate.hpp](#)

6.2 Ipopt::AlgorithmBuilder Class Reference

Builder to create a complete `IpoptAlg` object.

```
#include <IpAlgBuilder.hpp>
```

Inheritance diagram for `Ipopt::AlgorithmBuilder`:



Public Member Functions

Constructors/Destructors

- `AlgorithmBuilder (SmartPointer< AugSystemSolver > custom_solver=NULL)`
Constructor.
- `virtual ~AlgorithmBuilder ()`
Destructor.

Methods to build parts of the algorithm

- `virtual void BuildIpoptObjects (const Journalist &jnlst, const OptionsList &options, const std::string &prefix, const SmartPtr< NLP > &nlp, SmartPtr< IpoptNLP > &ip_nlp, SmartPtr< IpoptData > &ip_data, SmartPtr< IpoptCalculatedQuantities > &ip_cq)`

- virtual `SmartPtr< IpoptAlgorithm > BuildBasicAlgorithm` (const `Journalist` &jnlst, const `OptionsList` &options, const std::string &prefix)

Static Public Member Functions

- static void `RegisterOptions` (`SmartPtr< RegisteredOptions >` roptions)

Methods for IpoptTypeInfo.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `AlgorithmBuilder` (const `AlgorithmBuilder` &)
Default Constructor.
- void `operator=` (const `AlgorithmBuilder` &)
Overloaded Equals Operator.

Private Attributes

- `SmartPtr< AugSystemSolver > custom_solver_`
Optional pointer to AugSystemSolver.

6.2.1 Detailed Description

Builder to create a complete IpoptAlg object.

This object contains all subelements (such as line search objects etc). How the resulting IpoptAlg object is built can be influenced by the options.

The optional argument custom_solver allows the expert user to provide a specialized linear solver (e.g., of the type `GenAugSystemSolver`), possibly for selfmade matrix objects.

TODO: Currently, this is a basic implementation with everything in one method that can be overloaded. This will need to be expanded to allow customization of different parts without recoding everything.

Definition at line 32 of file IpAlgBuilder.hpp.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `Ipopt::AlgorithmBuilder::AlgorithmBuilder (SmartPtr< AugSystemSolver > custom_solver = NULL)`

Constructor.

6.2.2.2 `virtual Ipopt::AlgorithmBuilder::~~AlgorithmBuilder () [inline], [virtual]`

Destructor.

Definition at line 41 of file IpAlgBuilder.hpp.

6.2.2.3 Ipopt::AlgorithmBuilder::AlgorithmBuilder (const AlgorithmBuilder &) [private]

Default Constructor.

Copy Constructor

6.2.3 Member Function Documentation

6.2.3.1 virtual void Ipopt::AlgorithmBuilder::BuildIpoptObjects (const Journalist & *jnlst*, const OptionsList & *options*, const std::string & *prefix*, const SmartPtr< NLP > & *nlp*, SmartPtr< IpoptNLP > & *ip_nlp*, SmartPtr< IpoptData > & *ip_data*, SmartPtr< IpoptCalculatedQuantities > & *ip_cq*) [virtual]

Reimplemented in [Ipopt::InexactAlgorithmBuilder](#).

6.2.3.2 virtual SmartPtr<IpoptAlgorithm> Ipopt::AlgorithmBuilder::BuildBasicAlgorithm (const Journalist & *jnlst*, const OptionsList & *options*, const std::string & *prefix*) [virtual]

Reimplemented in [Ipopt::InexactAlgorithmBuilder](#).

6.2.3.3 static void Ipopt::AlgorithmBuilder::RegisterOptions (SmartPtr< RegisteredOptions > *roptions*) [static]

Methods for IpoptTypeInfo.

register the options used by the algorithm builder

6.2.3.4 void Ipopt::AlgorithmBuilder::operator= (const AlgorithmBuilder &) [private]

Overloaded Equals Operator.

6.2.4 Member Data Documentation

6.2.4.1 SmartPtr<AugSystemSolver> Ipopt::AlgorithmBuilder::custom_solver_ [private]

Optional pointer to [AugSystemSolver](#).

If this is set in the constructor, we will use this to solve the linear systems if the option linear_solver=custerm is chosen.

Definition at line 89 of file IpAlgBuilder.hpp.

The documentation for this class was generated from the following file:

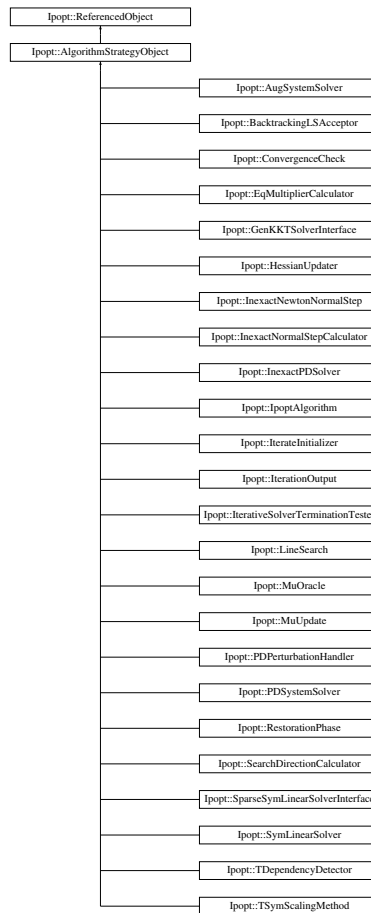
- [Algorithm/IpAlgBuilder.hpp](#)

6.3 Ipopt::AlgorithmStrategyObject Class Reference

This is the base class for all algorithm strategy objects.

```
#include <IpAlgStrategy.hpp>
```

Inheritance diagram for Ipopt::AlgorithmStrategyObject:



Public Member Functions

- bool **Initialize** (const **Journalist** &jnlst, **IpoptNLP** &ip_nlp, **IpoptData** &ip_data, **IpoptCalculatedQuantities** &ip_cq, const **OptionsList** &options, const std::string &prefix)
This method is called every time the algorithm starts again - it is used to reset any internal state.
- bool **ReducedInitialize** (const **Journalist** &jnlst, const **OptionsList** &options, const std::string &prefix)
*Reduced version of the Initialize method, which does not require special **Ipopt** information.*

Constructors/Destructors

- **AlgorithmStrategyObject** ()
Default Constructor.
- virtual **~AlgorithmStrategyObject** ()
Default Destructor.

Protected Member Functions

- virtual bool **InitializeImpl** (const **OptionsList** &options, const std::string &prefix)=0
Implementation of the initialization method that has to be overloaded by for each derived class.

Accessor methods for the problem defining objects.

Those should be used by the derived classes.

- const [Journalist](#) & [Jnlst](#) () const
- [IpoptNLP](#) & [IpNLP](#) () const
- [IpoptData](#) & [IpData](#) () const
- [IpoptCalculatedQuantities](#) & [IpCq](#) () const
- bool [HaveIpData](#) () const

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [AlgorithmStrategyObject](#) (const [AlgorithmStrategyObject](#) &)
Default Constructor.
- void [operator=](#) (const [AlgorithmStrategyObject](#) &)
Overloaded Equals Operator.

Private Attributes

- bool [initialize_called_](#)
flag indicating if Initialize method has been called (for debugging)

Pointers to objects defining a particular optimization

problem

- [SmartPtr](#)< const [Journalist](#) > [jnlst_](#)
- [SmartPtr](#)< [IpoptNLP](#) > [ip_nlp_](#)
- [SmartPtr](#)< [IpoptData](#) > [ip_data_](#)
- [SmartPtr](#)
< [IpoptCalculatedQuantities](#) > [ip_cq_](#)

6.3.1 Detailed Description

This is the base class for all algorithm strategy objects.

The [AlgorithmStrategyObject](#) base class implements a common interface for all algorithm strategy objects. A strategy object is a component of the algorithm for which different alternatives or implementations exists. It allows to compose the algorithm before execution for a particular configuration, without the need to call alternatives based on enums. For example, the [LineSearch](#) object is a strategy object, since different line search options might be used for different runs.

This interface is used for things that are done to all strategy objects, like initialization and setting options.

Definition at line 35 of file [IpAlgStrategy.hpp](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Ipopt::AlgorithmStrategyObject::AlgorithmStrategyObject () [\[inline\]](#)

Default Constructor.

Definition at line 41 of file [IpAlgStrategy.hpp](#).

6.3.2.2 `virtual Ipopt::AlgorithmStrategyObject::~~AlgorithmStrategyObject () [inline],[virtual]`

Default Destructor.

Definition at line 47 of file IpAlgStrategy.hpp.

6.3.2.3 `Ipopt::AlgorithmStrategyObject::AlgorithmStrategyObject (const AlgorithmStrategyObject &) [private]`

Default Constructor.

Copy Constructor

6.3.3 Member Function Documentation

6.3.3.1 `bool Ipopt::AlgorithmStrategyObject::Initialize (const Journalist & jnlst, IpoptNLP & ip_nlp, IpoptData & ip_data, IpoptCalculatedQuantities & ip_cq, const OptionsList & options, const std::string & prefix) [inline]`

This method is called every time the algorithm starts again - it is used to reset any internal state.

The pointers to the [Journalist](#), as well as to the [IpoptNLP](#), [IpoptData](#), and [IpoptCalculatedQuantities](#) objects should be stored in the instantiation of this base class. This method is also used to get all required user options from the [OptionsList](#). Here, if prefix is given, each tag (identifying the options) is first looked for with the prefix in front, and if not found, without the prefix. Note: you should not cue off of the iteration count to indicate the "start" of an algorithm!

Do not overload this method, since it does some general initialization that is common for all strategy objects. Overload the protected `InitializeImpl` method instead.

Definition at line 66 of file IpAlgStrategy.hpp.

6.3.3.2 `bool Ipopt::AlgorithmStrategyObject::ReducedInitialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [inline]`

Reduced version of the `Initialize` method, which does not require special [Ipopt](#) information.

This is useful for algorithm objects that could be used outside [Ipopt](#), such as linear solvers.

Definition at line 92 of file IpAlgStrategy.hpp.

6.3.3.3 `virtual bool Ipopt::AlgorithmStrategyObject::InitializeImpl (const OptionsList & options, const std::string & prefix) [protected],[pure virtual]`

Implementation of the initialization method that has to be overloaded by for each derived class.

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::SparseSymLinearSolverInterface](#), [Ipopt::PDSystemSolver](#), [Ipopt::IpoptAlgorithm](#), [Ipopt::SymLinearSolver](#), [Ipopt::BacktrackingLineSearch](#), [Ipopt::IterativeSolverTerminationTester](#), [Ipopt::AugSystemSolver](#), [Ipopt::TSymLinearSolver](#), [Ipopt::PDFullSpaceSolver](#), [Ipopt::RestorationPhase](#), [Ipopt::ConvergenceCheck](#), [Ipopt::DefaultIterateInitializer](#), [Ipopt::RestoFilterConvergenceCheck](#), [Ipopt::RestoPenaltyConvergenceCheck](#), [Ipopt::RestoConvergenceCheck](#), [Ipopt::AugRestoSystemSolver](#), [Ipopt::LowRankSSAugSystemSolver](#), [Ipopt::MinC_1NrmRestorationPhase](#), [Ipopt::InexactPDSolver](#), [Ipopt::StdAugSystemSolver](#), [Ipopt::Ma57TSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::LimMemQuasiNewtonUpdater](#), [Ipopt::RestoIterateInitializer](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::InexactSearchDirCalculator](#), [Ipopt::FilterLSAcceptor](#), [Ipopt::PenaltyLSAcceptor](#), [Ipopt::RestoIterationOutput](#), [Ipopt::RestoRestorationPhase](#), [Ipopt::CGPenaltyLSAcceptor](#), [Ipopt::CGPerturbationHandler](#), [Ipopt::CGSearchDirCalculator](#), [Ipopt::InexactLSAcceptor](#), [Ipopt::InexactTSymScalingMethod](#), [Ipopt::AdaptiveMuUpdate](#), [Ipopt::HessianUpdater](#), [Ipopt::IterateInitializer](#), [Ipopt::IterationOutput](#), [Ipopt::LeastSquareMultipliers](#), [Ipopt::LowRankAugSystemSolver](#), [Ipopt::PDPerturbationHandler](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmSolverInterface](#), [Ipopt::BacktrackingLSAcceptor](#), [Ipopt::IterativeWsmSolverInterface](#), [Ipopt::SlackBasedTSymScalingMethod](#), [Ipopt::TSymScalingMethod](#), [Ipopt::InexactNormalStepCalculator](#), [Ipopt::EqMultiplierCalculator](#), [Ipopt::ExactHessianUpdater](#), [Ipopt::GenAugSystemSolver](#), [Ipopt::MonotoneMuUpdate](#), [Ipopt::MuOracle](#), [Ipopt::MuUpdate](#), [Ipopt::SearchDirectionCalculator](#), [Ipopt::InexactDoglegNormalStep](#), [Ipopt::Inexact](#)

[NewtonNormalStep](#), [Ipopt::InexactNormalTerminationTester](#), [Ipopt::InexactPDTerminationTester](#), [Ipopt::PDSearchDir-Calculator](#), [Ipopt::QualityFunctionMuOracle](#), [Ipopt::WarmStartIterateInitializer](#), [Ipopt::Mc19TSymScalingMethod](#), [Ipopt::OptimalityErrorConvergenceCheck](#), [Ipopt::ProbingMuOracle](#), [Ipopt::GenKKTSolverInterface](#), [Ipopt::TDependency-Detector](#), [Ipopt::LoqoMuOracle](#), [Ipopt::OrigIterationOutput](#), [Ipopt::Ma27TSolverInterface](#), [Ipopt::TSymDependency-Detector](#), and [Ipopt::Ma28TDependencyDetector](#).

6.3.3.4 `const Journalist& Ipopt::AlgorithmStrategyObject::Jnlst () const` `[inline],[protected]`

Definition at line 120 of file `IpAlgStrategy.hpp`.

6.3.3.5 `IpoptNLP& Ipopt::AlgorithmStrategyObject::IpNLP () const` `[inline],[protected]`

Definition at line 125 of file `IpAlgStrategy.hpp`.

6.3.3.6 `IpoptData& Ipopt::AlgorithmStrategyObject::IpData () const` `[inline],[protected]`

Definition at line 131 of file `IpAlgStrategy.hpp`.

6.3.3.7 `IpoptCalculatedQuantities& Ipopt::AlgorithmStrategyObject::IpCq () const` `[inline],[protected]`

Definition at line 137 of file `IpAlgStrategy.hpp`.

6.3.3.8 `bool Ipopt::AlgorithmStrategyObject::HaveIpData () const` `[inline],[protected]`

Definition at line 143 of file `IpAlgStrategy.hpp`.

6.3.3.9 `void Ipopt::AlgorithmStrategyObject::operator= (const AlgorithmStrategyObject &)` `[private]`

Overloaded Equals Operator.

6.3.4 Member Data Documentation

6.3.4.1 `SmartPtr<const Journalist> Ipopt::AlgorithmStrategyObject::jnlst_` `[private]`

Definition at line 172 of file `IpAlgStrategy.hpp`.

6.3.4.2 `SmartPtr<IpoptNLP> Ipopt::AlgorithmStrategyObject::ip_nlp_` `[private]`

Definition at line 173 of file `IpAlgStrategy.hpp`.

6.3.4.3 `SmartPtr<IpoptData> Ipopt::AlgorithmStrategyObject::ip_data_` `[private]`

Definition at line 174 of file `IpAlgStrategy.hpp`.

6.3.4.4 `SmartPtr<IpoptCalculatedQuantities> Ipopt::AlgorithmStrategyObject::ip_cq_` `[private]`

Definition at line 175 of file `IpAlgStrategy.hpp`.

6.3.4.5 `bool Ipopt::AlgorithmStrategyObject::initialize_called_` `[private]`

flag indicating if Initialize method has been called (for debugging)

Definition at line 180 of file `IpAlgStrategy.hpp`.

The documentation for this class was generated from the following file:

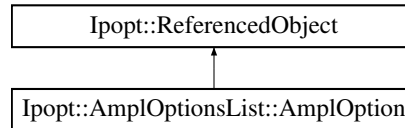
- [Algorithm/IpAlgStrategy.hpp](#)

6.4 Ipopt::AmplOptionsList::AmplOption Class Reference

Ampl Option class, contains name, type and description for an AMPL option.

```
#include <AmplTNLP.hpp>
```

Inheritance diagram for Ipopt::AmplOptionsList::AmplOption:



Public Member Functions

- [AmplOption](#) (const std::string ipopt_option_name, [AmplOptionType](#) type, const std::string description)
- [~AmplOption](#) ()
- const std::string & [IpoptOptionName](#) () const
- [AmplOptionType](#) [Type](#) () const
- char * [Description](#) () const

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [AmplOption](#) ()
Default Constructor.
- [AmplOption](#) (const [AmplOption](#) &)
Copy Constructor.
- void [operator=](#) (const [AmplOption](#) &)
Overloaded Equals Operator.

Private Attributes

- const std::string [ipopt_option_name_](#)
- const [AmplOptionType](#) [type_](#)
- char * [description_](#)

6.4.1 Detailed Description

Ampl Option class, contains name, type and description for an AMPL option.

Definition at line 115 of file AmplTNLP.hpp.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 `Ipopt::AmplOptionsList::AmplOption::AmplOption (const std::string ipopt_option_name, AmplOptionType type, const std::string description)`

6.4.2.2 `Ipopt::AmplOptionsList::AmplOption::~~AmplOption ()` `[inline]`

Definition at line 122 of file `AmplTNLP.hpp`.

6.4.2.3 `Ipopt::AmplOptionsList::AmplOption::AmplOption ()` `[private]`

Default Constructor.

6.4.2.4 `Ipopt::AmplOptionsList::AmplOption::AmplOption (const AmplOption &)` `[private]`

Copy Constructor.

6.4.3 Member Function Documentation

6.4.3.1 `const std::string& Ipopt::AmplOptionsList::AmplOption::IpoptOptionName () const` `[inline]`

Definition at line 127 of file `AmplTNLP.hpp`.

6.4.3.2 `AmplOptionType Ipopt::AmplOptionsList::AmplOption::Type () const` `[inline]`

Definition at line 131 of file `AmplTNLP.hpp`.

6.4.3.3 `char* Ipopt::AmplOptionsList::AmplOption::Description () const` `[inline]`

Definition at line 135 of file `AmplTNLP.hpp`.

6.4.3.4 `void Ipopt::AmplOptionsList::AmplOption::operator= (const AmplOption &)` `[private]`

Overloaded Equals Operator.

6.4.4 Member Data Documentation

6.4.4.1 `const std::string Ipopt::AmplOptionsList::AmplOption::ipopt_option_name_` `[private]`

Definition at line 158 of file `AmplTNLP.hpp`.

6.4.4.2 `const AmplOptionType Ipopt::AmplOptionsList::AmplOption::type_` `[private]`

Definition at line 159 of file `AmplTNLP.hpp`.

6.4.4.3 `char* Ipopt::AmplOptionsList::AmplOption::description_` `[private]`

Definition at line 160 of file `AmplTNLP.hpp`.

The documentation for this class was generated from the following file:

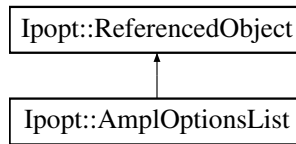
- `Apps/AmplSolver/AmplTNLP.hpp`

6.5 Ipopt::AmplOptionsList Class Reference

Class for storing a number of AMPL options that should be registered to the AMPL Solver library interface.

```
#include <AmplTNLP.hpp>
```

Inheritance diagram for Ipopt::AmplOptionsList:



Classes

- class [AmplOption](#)
Ampl Option class, contains name, type and description for an AMPL option.
- class [PrivatInfo](#)

Public Types

- enum [AmplOptionType](#) {
 [String_Option](#), [Number_Option](#), [Integer_Option](#), [WS_Option](#),
 [HaltOnError_Option](#) }

Public Member Functions

- [AmplOptionsList](#) ()
Default Constructor.
- [~AmplOptionsList](#) ()
Destructor.
- void [AddAmplOption](#) (const std::string ampl_option_name, const std::string ipopt_option_name, [AmplOptionsList::AmplOptionType](#) type, const std::string description)
Adding a new AMPL Option.
- [Index NumberOfAmplOptions](#) ()
Number of AMPL Options.
- void * [Keywords](#) (const [SmartPtr](#)< [OptionsList](#) > &options, [SmartPtr](#)< const [Journalist](#) > jnlst, void **error)
ASL keywords list for the stored options.

Private Member Functions

- void [MakeValidLatexString](#) (std::string source, std::string &dest) const
- void [PrintLatex](#) ([SmartPtr](#)< const [Journalist](#) > jnlst)

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [AmplOptionsList](#) (const [AmplOptionsList](#) &)
Default Constructor.
- void [operator=](#) (const [AmplOptionsList](#) &)
Overloaded Equals Operator.

Private Attributes

- std::map< std::string,
[SmartPtr](#)< const [AmplOption](#) > > [ampl_options_map_](#)
map for storing registered AMPL options
- void * [keywds_](#)
pointer to the keywords
- [Index](#) [nkeywds_](#)
Number of entries stored in keywds_.

6.5.1 Detailed Description

Class for storing a number of AMPL options that should be registered to the AMPL Solver library interface.
Definition at line 102 of file `AmplTNLP.hpp`.

6.5.2 Member Enumeration Documentation

6.5.2.1 enum Ipopt::AmplOptionsList::AmplOptionType

Enumerator

String_Option
Number_Option
Integer_Option
WS_Option
HaltOnError_Option

Definition at line 105 of file `AmplTNLP.hpp`.

6.5.3 Constructor & Destructor Documentation

6.5.3.1 Ipopt::AmplOptionsList::AmplOptionsList () [inline]

Default Constructor.

Definition at line 201 of file `AmplTNLP.hpp`.

6.5.3.2 Ipopt::AmplOptionsList::~AmplOptionsList ()

Destructor.

6.5.3.3 Ipopt::AmplOptionsList::AmplOptionsList (const [AmplOptionsList](#) &) [private]

Default Constructor.

Copy Constructor

6.5.4 Member Function Documentation

6.5.4.1 `void Ipopt::AmplOptionsList::AddAmplOption (const std::string ampl_option_name, const std::string ipopt_option_name, AmplOptionsList::AmplOptionType type, const std::string description) [inline]`

Adding a new AMPL Option.

Definition at line 211 of file AmplTNLP.hpp.

6.5.4.2 `Index Ipopt::AmplOptionsList::NumberOfAmplOptions () [inline]`

Number of AMPL Options.

Definition at line 222 of file AmplTNLP.hpp.

6.5.4.3 `void* Ipopt::AmplOptionsList::Keywords (const SmartPtr< OptionsList > & options, SmartPtr< const Journalist > jnlst, void ** error)`

ASL keywords list for the stored options.

6.5.4.4 `void Ipopt::AmplOptionsList::operator= (const AmplOptionsList &) [private]`

Overloaded Equals Operator.

6.5.4.5 `void Ipopt::AmplOptionsList::MakeValidLatexString (std::string source, std::string & dest) const [private]`

6.5.4.6 `void Ipopt::AmplOptionsList::PrintLatex (SmartPtr< const Journalist > jnlst) [private]`

6.5.5 Member Data Documentation

6.5.5.1 `std::map<std::string, SmartPtr<const AmplOption>> Ipopt::AmplOptionsList::ampl_options_map_ [private]`

map for storing registered AMPL options

Definition at line 256 of file AmplTNLP.hpp.

6.5.5.2 `void* Ipopt::AmplOptionsList::keywds_ [private]`

pointer to the keywords

Definition at line 262 of file AmplTNLP.hpp.

6.5.5.3 `Index Ipopt::AmplOptionsList::nkeywds_ [private]`

Number of entries stored in keywds_.

Definition at line 265 of file AmplTNLP.hpp.

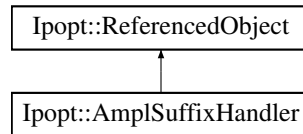
The documentation for this class was generated from the following file:

- Apps/AmplSolver/[AmplTNLP.hpp](#)

6.6 Ipopt::AmplSuffixHandler Class Reference

```
#include <AmplTNLP.hpp>
```

Inheritance diagram for Ipopt::AmplSuffixHandler:



Public Types

- enum [Suffix_Type](#) { [Index_Type](#), [Number_Type](#) }
- enum [Suffix_Source](#) { [Variable_Source](#), [Constraint_Source](#), [Objective_Source](#), [Problem_Source](#) }

Public Member Functions

- [AmplSuffixHandler](#) ()
- [~AmplSuffixHandler](#) ()
- void [AddAvailableSuffix](#) (std::string suffix_string, [Suffix_Source](#) source, [Suffix_Type](#) type)
- const [Index](#) * [GetIntegerSuffixValues](#) (std::string suffix_string, [Suffix_Source](#) source) const
- const [Number](#) * [GetNumberSuffixValues](#) (std::string suffix_string, [Suffix_Source](#) source) const
- std::vector< [Index](#) > [GetIntegerSuffixValues](#) ([Index](#) n, std::string suffix_string, [Suffix_Source](#) source) const
- std::vector< [Number](#) > [GetNumberSuffixValues](#) ([Index](#) n, std::string suffix_string, [Suffix_Source](#) source) const

Private Member Functions

- void [PrepareAmplForSuffixes](#) (ASL_pfgh *asl)
Method called by [AmplTNLP](#) to prepare the asl for the suffixes.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [AmplSuffixHandler](#) (const [AmplSuffixHandler](#) &)
Default Constructor.
- void [operator=](#) (const [AmplSuffixHandler](#) &)
Overloaded Equals Operator.

Private Attributes

- ASL_pfgh * [asl_](#)
- SufDecl * [sufstab_](#)
- std::vector< std::string > [suffix_ids_](#)
- std::vector< [Suffix_Type](#) > [suffix_types_](#)
- std::vector< [Suffix_Source](#) > [suffix_sources_](#)

Friends

- class [AmplTNLP](#)
Method called by [AmplTNLP](#) to retrieve the suffixes from asl.

6.6.1 Detailed Description

Definition at line 27 of file AmplTNLP.hpp.

6.6.2 Member Enumeration Documentation

6.6.2.1 enum Ipopt::AmplSuffixHandler::Suffix_Type

Enumerator

Index_Type

Number_Type

Definition at line 34 of file AmplTNLP.hpp.

6.6.2.2 enum Ipopt::AmplSuffixHandler::Suffix_Source

Enumerator

Variable_Source

Constraint_Source

Objective_Source

Problem_Source

Definition at line 40 of file AmplTNLP.hpp.

6.6.3 Constructor & Destructor Documentation

6.6.3.1 Ipopt::AmplSuffixHandler::AmplSuffixHandler ()

6.6.3.2 Ipopt::AmplSuffixHandler::~~AmplSuffixHandler ()

6.6.3.3 Ipopt::AmplSuffixHandler::AmplSuffixHandler (const AmplSuffixHandler &) [private]

Default Constructor.

Copy Constructor

6.6.4 Member Function Documentation

6.6.4.1 void Ipopt::AmplSuffixHandler::AddAvailableSuffix (std::string *suffix_string*, Suffix_Source *source*, Suffix_Type *type*) [inline]

Definition at line 48 of file AmplTNLP.hpp.

6.6.4.2 const Index* Ipopt::AmplSuffixHandler::GetIntegerSuffixValues (std::string *suffix_string*, Suffix_Source *source*) const

6.6.4.3 const Number* Ipopt::AmplSuffixHandler::GetNumberSuffixValues (std::string *suffix_string*, Suffix_Source *source*) const

6.6.4.4 std::vector<Index> Ipopt::AmplSuffixHandler::GetIntegerSuffixValues (Index *n*, std::string *suffix_string*, Suffix_Source *source*) const

6.6.4.5 `std::vector<Number> Ipopt::AmplSuffixHandler::GetNumberSuffixValues (Index n, std::string suffix_string, Suffix_Source source) const`

6.6.4.6 `void Ipopt::AmplSuffixHandler::operator= (const AmplSuffixHandler &) [private]`

Overloaded Equals Operator.

6.6.4.7 `void Ipopt::AmplSuffixHandler::PrepareAmplForSuffixes (ASL_pfgh * asl) [private]`

Method called by [AmplTNLP](#) to prepare the asl for the suffixes.

6.6.5 Friends And Related Function Documentation

6.6.5.1 `friend class AmplTNLP [friend]`

Method called by [AmplTNLP](#) to retrieve the suffixes from asl.

Definition at line 97 of file AmplTNLP.hpp.

6.6.6 Member Data Documentation

6.6.6.1 `ASL_pfgh* Ipopt::AmplSuffixHandler::asl_ [mutable],[private]`

Definition at line 83 of file AmplTNLP.hpp.

6.6.6.2 `SufDecl* Ipopt::AmplSuffixHandler::sufstab_ [private]`

Definition at line 85 of file AmplTNLP.hpp.

6.6.6.3 `std::vector<std::string> Ipopt::AmplSuffixHandler::suffix_ids_ [private]`

Definition at line 87 of file AmplTNLP.hpp.

6.6.6.4 `std::vector<Suffix_Type> Ipopt::AmplSuffixHandler::suffix_types_ [private]`

Definition at line 88 of file AmplTNLP.hpp.

6.6.6.5 `std::vector<Suffix_Source> Ipopt::AmplSuffixHandler::suffix_sources_ [private]`

Definition at line 89 of file AmplTNLP.hpp.

The documentation for this class was generated from the following file:

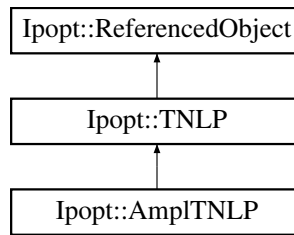
- Apps/AmplSolver/[AmplTNLP.hpp](#)

6.7 Ipopt::AmplTNLP Class Reference

Ampl Interface.

```
#include <AmplTNLP.hpp>
```

Inheritance diagram for Ipopt::AmplTNLP:



Public Member Functions

- [DECLARE_STD_EXCEPTION](#) (NONPOSITIVE_SCALING_FACTOR)

Exceptions.

- void [set_active_objective](#) (Index obj_no)

A method for setting the index of the objective function to be considered.

- [SmartPtr](#)< [AmplSuffixHandler](#) > [get_suffix_handler](#) ()

Method for returning the suffix handler.

Constructors/Destructors

- [AmplTNLP](#) (const [SmartPtr](#)< const [Journalist](#) > &jnlst, const [SmartPtr](#)< [OptionsList](#) > options, char **&argv, [SmartPtr](#)< [AmplSuffixHandler](#) > suffix_handler=NULL, bool allow_discrete=false, [SmartPtr](#)< [AmplOptionsList](#) > ampl_options_list=NULL, const char *ampl_option_string=NULL, const char *ampl_invocation_string=NULL, const char *ampl_banner_string=NULL, std::string *nl_file_content=NULL)

Constructor.

- virtual [~AmplTNLP](#) ()

Default destructor.

methods to gather information about the NLP. These

methods are overloaded from [TNLP](#).

See [TNLP](#) for their more detailed documentation.

- virtual bool [get_nlp_info](#) (Index &n, Index &m, Index &nnz_jac_g, Index &nnz_h_lag, IndexStyleEnum &index_style)
returns dimensions of the nlp.
- virtual bool [get_var_con_metadata](#) (Index n, [StringMetaDataMapType](#) &var_string_md, [IntegerMetaDataMapType](#) &var_integer_md, [NumericMetaDataMapType](#) &var_numeric_md, Index m, [StringMetaDataMapType](#) &con_string_md, [IntegerMetaDataMapType](#) &con_integer_md, [NumericMetaDataMapType](#) &con_numeric_md)
returns names and other meta data for the variables and constraints Overloaded from [TNLP](#)
- virtual bool [get_bounds_info](#) (Index n, Number *x_l, Number *x_u, Index m, Number *g_l, Number *g_u)
returns bounds of the nlp.
- virtual bool [get_constraints_linearity](#) (Index m, [LinearityType](#) *const_types)
Returns the constraint linearity.
- virtual bool [get_starting_point](#) (Index n, bool init_x, Number *x, bool init_z, Number *z_L, Number *z_U, Index m, bool init_lambda, Number *lambda)
provides a starting point for the nlp variables.
- virtual bool [eval_f](#) (Index n, const Number *x, bool new_x, Number &obj_value)
evaluates the objective value for the nlp.
- virtual bool [eval_grad_f](#) (Index n, const Number *x, bool new_x, Number *grad_f)
evaluates the gradient of the objective for the nlp.

- virtual bool `eval_g` (Index n, const Number *x, bool new_x, Index m, Number *g)
evaluates the constraint residuals for the nlp.
- virtual bool `eval_jac_g` (Index n, const Number *x, bool new_x, Index m, Index nele_jac, Index *iRow, Index *jCol, Number *values)
specifies the jacobian structure (if values is NULL) and evaluates the jacobian values (if values is not NULL) for the nlp.
- virtual bool `eval_h` (Index n, const Number *x, bool new_x, Number obj_factor, Index m, const Number *lambda, bool new_lambda, Index nele_hess, Index *iRow, Index *jCol, Number *values)
specifies the structure of the hessian of the lagrangian (if values is NULL) and evaluates the values (if values is not NULL).
- virtual bool `get_scaling_parameters` (Number &obj_scaling, bool &use_x_scaling, Index n, Number *x_scaling, bool &use_g_scaling, Index m, Number *g_scaling)
retrieve the scaling parameters for the variables, objective function, and constraints.

Solution Methods

- virtual void `finalize_solution` (SolverReturn status, Index n, const Number *x, const Number *z_L, const Number *z_U, Index m, const Number *g, const Number *lambda, Number obj_value, const IpoptData *ip_data, IpoptCalculatedQuantities *ip_cq)
This method is called when the algorithm is complete so the [TNLP](#) can store/write the solution.

Method for quasi-Newton approximation information.

- virtual Index `get_number_of_nonlinear_variables` ()
- virtual bool `get_list_of_nonlinear_variables` (Index num_nonlin_vars, Index *pos_nonlin_vars)

Ampl specific methods

- ASL_pfgh * `AmplSolverObject` ()
Return the ampl solver object (ASL)*
- void `write_solution_file` (const std::string &message) const
Write the solution file.
- void `get_discrete_info` (Index &nlvb_, Index &nlvbi_, Index &nlvc_, Index &nlvci_, Index &nlvo_, Index &nlvoi_, Index &nbv_, Index &niv_) const
ampl orders the variables like (continuous, binary, integer).

Methods to set meta data for the variables

and constraints.

These values will be passed on to the [TNLP](#) in `get_var_con_meta_data`

- void `set_string_metadata_for_var` (std::string tag, std::vector< std::string > meta_data)
- void `set_integer_metadata_for_var` (std::string tag, std::vector< Index > meta_data)
- void `set_numeric_metadata_for_var` (std::string tag, std::vector< Number > meta_data)
- void `set_string_metadata_for_con` (std::string tag, std::vector< std::string > meta_data)
- void `set_integer_metadata_for_con` (std::string tag, std::vector< Index > meta_data)
- void `set_numeric_metadata_for_con` (std::string tag, std::vector< Number > meta_data)

Private Member Functions

- bool `internal_objval` (const Number *x, Number &obj_val)
Make the objective call to ampl.
- bool `internal_conval` (const Number *x, Index m, Number *g=NULL)
Make the constraint call to ampl.
- bool `apply_new_x` (bool new_x, Index n, const Number *x)

Internal function to update the internal and ampl state if the x value changes.

- char * [get_options](#) (const [SmartPtr](#)< [OptionsList](#) > &options, [SmartPtr](#)< [AmplOptionsList](#) > &l_options_list, const char *ampl_option_string, const char *ampl_invokation_string, const char *ampl_banner_string, char **&argv)

Method for obtaining the name of the NL file and the options set from AMPL.

- bool [nerror_ok](#) (void *nerror)
returns true if the ampl nerror code is ok
- void [call_heset](#) ()
calls heset ASL function

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [AmplTNLP](#) ()
Default Constructor.
- [AmplTNLP](#) (const [AmplTNLP](#) &)
Copy Constructor.
- void [operator=](#) (const [AmplTNLP](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< const [Journalist](#) > [jnlst_](#)
Journalist.
- ASL_pfgh * [asl_](#)
pointer to the main ASL structure
- double [obj_sign_](#)
Sign of the objective fn (1 for min, -1 for max)
- void * [Oinfo_ptr_](#)
Pointer to the Oinfo structure.
- void * [nerror_](#)
nerror flag passed to ampl calls - set to NULL to halt on error
- [SmartPtr](#)< [AmplSuffixHandler](#) > [suffix_handler_](#)
Suffix Handler.
- [StringMetaDataMapType](#) [var_string_md_](#)
meta data to pass on to TNLP
- [IntegerMetaDataMapType](#) [var_integer_md_](#)
- [NumericMetaDataMapType](#) [var_numeric_md_](#)
- [StringMetaDataMapType](#) [con_string_md_](#)
- [IntegerMetaDataMapType](#) [con_integer_md_](#)
- [NumericMetaDataMapType](#) [con_numeric_md_](#)

Problem Size Data

- [Index nz_h_full_](#)

Internal copies of data

- [Number * x_sol_](#)
Solution Vectors.
- [Number * z_L_sol_](#)
- [Number * z_U_sol_](#)
- [Number * g_sol_](#)
- [Number * lambda_sol_](#)
- [Number obj_sol_](#)

Flags to track internal state

- [bool objval_called_with_current_x_](#)
true when the objective value has been calculated with the current x, set to false in apply_new_x, and set to true in internal_objval
- [bool conval_called_with_current_x_](#)
true when the constraint values have been calculated with the current x, set to false in apply_new_x, and set to true in internal_conval
- [bool hesset_called_](#)
true when we have called hesset
- [bool set_active_objective_called_](#)
true when set_active_objective has been called

Additional Inherited Members

6.7.1 Detailed Description

Ampl Interface.

Ampl Interface, implemented as a [TNLP](#).

Definition at line 271 of file AmplTNLP.hpp.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 `Ipopt::AmplTNLP::AmplTNLP (const SmartPtr< const Journalist > &jnlst, const SmartPtr< OptionsList > options, char **& argv, SmartPtr< AmplSuffixHandler > suffix_handler = NULL, bool allow_discrete = false, SmartPtr< AmplOptionsList > ampl_options_list = NULL, const char * ampl_option_string = NULL, const char * ampl_invokation_string = NULL, const char * ampl_banner_string = NULL, std::string * nl_file_content = NULL)`

Constructor.

6.7.2.2 `virtual Ipopt::AmplTNLP::~~AmplTNLP () [virtual]`

Default destructor.

6.7.2.3 `Ipopt::AmplTNLP::AmplTNLP () [private]`

Default Constructor.

6.7.2.4 `Ipopt::AmplTNLP::AmplTNLP (const AmplTNLP &) [private]`

Copy Constructor.

6.7.3 Member Function Documentation

6.7.3.1 `Ipopt::AmplTNLP::DECLARE_STD_EXCEPTION (NONPOSITIVE_SCALING_FACTOR)`

Exceptions.

6.7.3.2 `virtual bool Ipopt::AmplTNLP::get_nlp_info (Index & n, Index & m, Index & nnz_jac_g, Index & nnz_h_lag, IndexStyleEnum & index_style) [virtual]`

returns dimensions of the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.7.3.3 `virtual bool Ipopt::AmplTNLP::get_var_con_metadata (Index n, StringMetaDataMapType & var_string_md, IntegerMetaDataMapType & var_integer_md, NumericMetaDataMapType & var_numeric_md, Index m, StringMetaDataMapType & con_string_md, IntegerMetaDataMapType & con_integer_md, NumericMetaDataMapType & con_numeric_md) [virtual]`

returns names and other meta data for the variables and constraints Overloaded from [TNLP](#)

Reimplemented from [Ipopt::TNLP](#).

6.7.3.4 `virtual bool Ipopt::AmplTNLP::get_bounds_info (Index n, Number * x_l, Number * x_u, Index m, Number * g_l, Number * g_u) [virtual]`

returns bounds of the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.7.3.5 `virtual bool Ipopt::AmplTNLP::get_constraints_linearity (Index m, LinearityType * const_types) [virtual]`

Returns the constraint linearity.

array will be allocated with length n. (default implementation just return false and does not fill the array).

Reimplemented from [Ipopt::TNLP](#).

6.7.3.6 `virtual bool Ipopt::AmplTNLP::get_starting_point (Index n, bool init_x, Number * x, bool init_z, Number * z_L, Number * z_U, Index m, bool init_lambda, Number * lambda) [virtual]`

provides a starting point for the nlp variables.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.7.3.7 `virtual bool Ipopt::AmplTNLP::eval_f (Index n, const Number * x, bool new_x, Number & obj_value) [virtual]`

evaluates the objective value for the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.7.3.8 `virtual bool Ipopt::AmplTNLP::eval_grad_f (Index n, const Number * x, bool new_x, Number * grad_f) [virtual]`

evaluates the gradient of the objective for the

nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.7.3.9 `virtual bool Ipopt::AmplTNLP::eval_g (Index n, const Number * x, bool new_x, Index m, Number * g)`
[virtual]

evaluates the constraint residuals for the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.7.3.10 `virtual bool Ipopt::AmplTNLP::eval_jac_g (Index n, const Number * x, bool new_x, Index m, Index nele_jac, Index * iRow, Index * jCol, Number * values)` [virtual]

specifies the jacobian structure (if values is NULL) and evaluates the jacobian values (if values is not NULL) for the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.7.3.11 `virtual bool Ipopt::AmplTNLP::eval_h (Index n, const Number * x, bool new_x, Number obj_factor, Index m, const Number * lambda, bool new_lambda, Index nele_hess, Index * iRow, Index * jCol, Number * values)`
[virtual]

specifies the structure of the hessian of the lagrangian (if values is NULL) and evaluates the values (if values is not NULL).

Overloaded from [TNLP](#)

Reimplemented from [Ipopt::TNLP](#).

6.7.3.12 `virtual bool Ipopt::AmplTNLP::get_scaling_parameters (Number & obj_scaling, bool & use_x_scaling, Index n, Number * x_scaling, bool & use_g_scaling, Index m, Number * g_scaling)` [virtual]

retrieve the scaling parameters for the variables, objective function, and constraints.

Reimplemented from [Ipopt::TNLP](#).

6.7.3.13 `virtual void Ipopt::AmplTNLP::finalize_solution (SolverReturn status, Index n, const Number * x, const Number * z_L, const Number * z_U, Index m, const Number * g, const Number * lambda, Number obj_value, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq)` [virtual]

This method is called when the algorithm is complete so the [TNLP](#) can store/write the solution.

Implements [Ipopt::TNLP](#).

6.7.3.14 `virtual Index Ipopt::AmplTNLP::get_number_of_nonlinear_variables ()` [virtual]

Reimplemented from [Ipopt::TNLP](#).

6.7.3.15 `virtual bool Ipopt::AmplTNLP::get_list_of_nonlinear_variables (Index num_nonlin_vars, Index * pos_nonlin_vars)`
[virtual]

Reimplemented from [Ipopt::TNLP](#).

6.7.3.16 `ASL_pfgh* Ipopt::AmplTNLP::AmplSolverObject ()` [inline]

Return the ampl solver object (ASL*)

Definition at line 387 of file AmplTNLP.hpp.

6.7.3.17 `void lpopt::AmplTNLP::write_solution_file (const std::string & message) const`

Write the solution file.

This is a wrapper for AMPL's write_sol. TODO Maybe this should be at a different place, or collect the numbers itself?

6.7.3.18 `void lpopt::AmplTNLP::get_discrete_info (Index & nlvb_, Index & nlvbi_, Index & nlvc_, Index & nlvci_, Index & nlvo_, Index & nlvoi_, Index & nbv_, Index & niv_) const`

ampl orders the variables like (continuous, binary, integer).

This method gives the number of binary and integer variables. For details, see Tables 3 and 4 in "Hooking Your Solver to AMPL"

6.7.3.19 `void lpopt::AmplTNLP::set_active_objective (Index obj_no)`

A method for setting the index of the objective function to be considered.

This method must be called after the constructor, and before anything else is called. It can only be called once, and if there is more than one objective function in the AMPL model, it MUST be called.

6.7.3.20 `void lpopt::AmplTNLP::set_string_metadata_for_var (std::string tag, std::vector< std::string > meta_data)`
[inline]

Definition at line 424 of file AmplTNLP.hpp.

6.7.3.21 `void lpopt::AmplTNLP::set_integer_metadata_for_var (std::string tag, std::vector< Index > meta_data)` [inline]

Definition at line 429 of file AmplTNLP.hpp.

6.7.3.22 `void lpopt::AmplTNLP::set_numeric_metadata_for_var (std::string tag, std::vector< Number > meta_data)`
[inline]

Definition at line 434 of file AmplTNLP.hpp.

6.7.3.23 `void lpopt::AmplTNLP::set_string_metadata_for_con (std::string tag, std::vector< std::string > meta_data)`
[inline]

Definition at line 439 of file AmplTNLP.hpp.

6.7.3.24 `void lpopt::AmplTNLP::set_integer_metadata_for_con (std::string tag, std::vector< Index > meta_data)` [inline]

Definition at line 444 of file AmplTNLP.hpp.

6.7.3.25 `void lpopt::AmplTNLP::set_numeric_metadata_for_con (std::string tag, std::vector< Number > meta_data)`
[inline]

Definition at line 449 of file AmplTNLP.hpp.

6.7.3.26 `SmartPointer<AmplSuffixHandler> lpopt::AmplTNLP::get_suffix_handler ()` [inline]

Method for returning the suffix handler.

Definition at line 456 of file AmplTNLP.hpp.

6.7.3.27 `void Ipopt::AmplTNLP::operator=(const AmplTNLP &) [private]`

Overloaded Equals Operator.

6.7.3.28 `bool Ipopt::AmplTNLP::internal_objval(const Number * x, Number & obj_val) [private]`

Make the objective call to ampl.

6.7.3.29 `bool Ipopt::AmplTNLP::internal_conval(const Number * x, Index m, Number * g=NULL) [private]`

Make the constraint call to ampl.

6.7.3.30 `bool Ipopt::AmplTNLP::apply_new_x(bool new_x, Index n, const Number * x) [private]`

Internal function to update the internal and ampl state if the x value changes.

6.7.3.31 `char* Ipopt::AmplTNLP::get_options(const SmartPtr< OptionsList > & options, SmartPtr< AmplOptionsList > & ampl_options_list, const char * ampl_option_string, const char * ampl_invokation_string, const char * ampl_banner_string, char **& argv) [private]`

Method for obtaining the name of the NL file and the options set from AMPL.

Returns a pointer to a char* with the name of the stub

6.7.3.32 `bool Ipopt::AmplTNLP::nerror_ok(void * nerror) [private]`

returns true if the ampl error code is ok

6.7.3.33 `void Ipopt::AmplTNLP::call_heset() [private]`

calls heset ASL function

6.7.4 Member Data Documentation

6.7.4.1 `SmartPtr<const Journalist> Ipopt::AmplTNLP::jnlst_ [private]`

Journalist.

Definition at line 481 of file AmplTNLP.hpp.

6.7.4.2 `ASL_pfg* Ipopt::AmplTNLP::asl_ [private]`

pointer to the main ASL structure

Definition at line 484 of file AmplTNLP.hpp.

6.7.4.3 `double Ipopt::AmplTNLP::obj_sign_ [private]`

Sign of the objective fn (1 for min, -1 for max)

Definition at line 487 of file AmplTNLP.hpp.

6.7.4.4 `Index Ipopt::AmplTNLP::nz_h_full_ [private]`

Definition at line 491 of file AmplTNLP.hpp.

6.7.4.5 `Number* Ipopt::AmplTNLP::x_sol_ [private]`

Solution Vectors.

Definition at line 498 of file AmplTNLP.hpp.

6.7.4.6 `Number* Ipopt::AmplTNLP::z_L_sol_ [private]`

Definition at line 499 of file AmplTNLP.hpp.

6.7.4.7 `Number* Ipopt::AmplTNLP::z_U_sol_ [private]`

Definition at line 500 of file AmplTNLP.hpp.

6.7.4.8 `Number* Ipopt::AmplTNLP::g_sol_ [private]`

Definition at line 501 of file AmplTNLP.hpp.

6.7.4.9 `Number* Ipopt::AmplTNLP::lambda_sol_ [private]`

Definition at line 502 of file AmplTNLP.hpp.

6.7.4.10 `Number Ipopt::AmplTNLP::obj_sol_ [private]`

Definition at line 503 of file AmplTNLP.hpp.

6.7.4.11 `bool Ipopt::AmplTNLP::objval_called_with_current_x_ [private]`

true when the objective value has been calculated with the current x, set to false in `apply_new_x`, and set to true in `internal_objval`

Definition at line 511 of file AmplTNLP.hpp.

6.7.4.12 `bool Ipopt::AmplTNLP::conval_called_with_current_x_ [private]`

true when the constraint values have been calculated with the current x, set to false in `apply_new_x`, and set to true in `internal_conval`

Definition at line 515 of file AmplTNLP.hpp.

6.7.4.13 `bool Ipopt::AmplTNLP::hesset_called_ [private]`

true when we have called `hesset`

Definition at line 517 of file AmplTNLP.hpp.

6.7.4.14 `bool Ipopt::AmplTNLP::set_active_objective_called_ [private]`

true when `set_active_objective` has been called

Definition at line 519 of file AmplTNLP.hpp.

6.7.4.15 `void* Ipopt::AmplTNLP::Oinfo_ptr_ [private]`

Pointer to the `Oinfo` structure.

Definition at line 523 of file AmplTNLP.hpp.

6.7.4.16 `void* Ipopt::AmplTNLP::nerror_ [private]`

nerror flag passed to ampl calls - set to NULL to halt on error

Definition at line 526 of file AmplTNLP.hpp.

6.7.4.17 `SmartPointer<AmplSuffixHandler> Ipopt::AmplTNLP::suffix_handler_ [private]`

Suffix Handler.

Definition at line 529 of file AmplTNLP.hpp.

6.7.4.18 `StringMetaDataMapType Ipopt::AmplTNLP::var_string_md_ [private]`

meta data to pass on to [TNLP](#)

Definition at line 557 of file AmplTNLP.hpp.

6.7.4.19 `IntegerMetaDataMapType Ipopt::AmplTNLP::var_integer_md_ [private]`

Definition at line 558 of file AmplTNLP.hpp.

6.7.4.20 `NumericMetaDataMapType Ipopt::AmplTNLP::var_numeric_md_ [private]`

Definition at line 559 of file AmplTNLP.hpp.

6.7.4.21 `StringMetaDataMapType Ipopt::AmplTNLP::con_string_md_ [private]`

Definition at line 560 of file AmplTNLP.hpp.

6.7.4.22 `IntegerMetaDataMapType Ipopt::AmplTNLP::con_integer_md_ [private]`

Definition at line 561 of file AmplTNLP.hpp.

6.7.4.23 `NumericMetaDataMapType Ipopt::AmplTNLP::con_numeric_md_ [private]`

Definition at line 562 of file AmplTNLP.hpp.

The documentation for this class was generated from the following file:

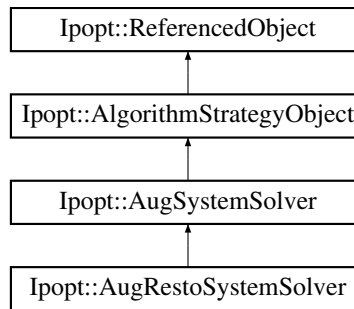
- Apps/AmplSolver/[AmplTNLP.hpp](#)

6.8 Ipopt::AugRestoSystemSolver Class Reference

Class that converts the an augmented system with compound restoration pieces into a smaller "pivoted" system to be solved with an existing [AugSystemSolver](#).

```
#include <IpAugRestoSystemSolver.hpp>
```

Inheritance diagram for Ipopt::AugRestoSystemSolver:



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual [ESymSolverStatus](#) [Solve](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) *J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) *J_d, const [Vector](#) *D_d, double delta_d, const [Vector](#) &rhs_x, const [Vector](#) &rhs_s, const [Vector](#) &rhs_c, const [Vector](#) &rhs_d, [Vector](#) &sol_x, [Vector](#) &sol_s, [Vector](#) &sol_c, [Vector](#) &sol_d, bool check_NegEVals, [Index](#) numberOfNegEVals)
Translate the augmented system (in the full space of the restoration variables) into the smaller space of the original variables.
- virtual [Index](#) [NumberOfNegEVals](#) () const
Returns the number of negative eigenvalues from the original augmented system call.
- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.

Constructors/Destructors

- [AugRestoSystemSolver](#) ([AugSystemSolver](#) &orig_aug_solver, bool skip_orig_aug_solver_init=true)
Constructor.
- virtual [~AugRestoSystemSolver](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [AugRestoSystemSolver](#) ()
Default Constructor.
- [AugRestoSystemSolver](#) (const [AugRestoSystemSolver](#) &)
Copy Constructor.
- void [operator=](#) (const [AugRestoSystemSolver](#) &)
Overloaded Equals Operator.

Methods to calculate the cached quantities

- [SmartPtr](#)< const [Vector](#) > [Neg_Omega_c_plus_D_c](#) (const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_n_c_inv, const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_p_c_inv, const [Vector](#) *D_c, const [Vector](#) &any_vec_in_c)
- [SmartPtr](#)< const [Vector](#) > [Neg_Omega_d_plus_D_d](#) (const [Matrix](#) &Pd_L, const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_n_d_inv, const [Matrix](#) &neg_Pd_U, const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_p_d_inv, const [Vector](#) *D_d, const [Vector](#) &any_vec_in_d)
- [SmartPtr](#)< const [Vector](#) > [Sigma_tilde_n_c_inv](#) (const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_n_c, [Number](#) delta_x, const [Vector](#) &any_vec_in_n_c)
Sigma tilde is the sum of Sigma and delta_x times the identity.
- [SmartPtr](#)< const [Vector](#) > [Sigma_tilde_p_c_inv](#) (const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_p_c, [Number](#) delta_x, const [Vector](#) &any_vec_in_p_c)
- [SmartPtr](#)< const [Vector](#) > [Sigma_tilde_n_d_inv](#) (const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_n_d, [Number](#) delta_x, const [Vector](#) &any_vec_in_n_d)
- [SmartPtr](#)< const [Vector](#) > [Sigma_tilde_p_d_inv](#) (const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_p_d, [Number](#) delta_x, const [Vector](#) &any_vec_in_p_d)
- [SmartPtr](#)< const [Vector](#) > [D_x_plus_wr_d](#) (const [SmartPtr](#)< const [Vector](#) > &CD_x0, [Number](#) factor, const [Vector](#) &wr_d)
- [SmartPtr](#)< const [Vector](#) > [Rhs_cR](#) (const [Vector](#) &rhs_c, const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_n_c_inv, const [Vector](#) &rhs_n_c, const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_p_c_inv, const [Vector](#) &rhs_p_c)
- [SmartPtr](#)< const [Vector](#) > [Rhs_dR](#) (const [Vector](#) &rhs_d, const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_n_d_inv, const [Vector](#) &rhs_n_d, const [Matrix](#) &pd_L, const [SmartPtr](#)< const [Vector](#) > &sigma_tilde_p_d_inv, const [Vector](#) &rhs_p_d, const [Matrix](#) &pd_U)

Private Attributes

- [SmartPtr](#)< [AugSystemSolver](#) > orig_aug_solver_
- bool skip_orig_aug_solver_init_

Caches for some of the necessary calculated quantities

- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > neg_omega_c_plus_D_c_cache_
- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > neg_omega_d_plus_D_d_cache_
- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > sigma_tilde_n_c_inv_cache_
- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > sigma_tilde_p_c_inv_cache_
- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > sigma_tilde_n_d_inv_cache_
- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > sigma_tilde_p_d_inv_cache_
- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > d_x_plus_wr_d_cache_
- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > rhs_cR_cache_
- [CachedResults](#)< [SmartPtr](#)< [Vector](#) > > rhs_dR_cache_

Additional Inherited Members

6.8.1 Detailed Description

Class that converts the an augmented system with compound restoration pieces into a smaller "pivoted" system to be solved with an existing [AugSystemSolver](#).

This is really a decorator that changes the behavior of the [AugSystemSolver](#) to account for the known structure of the restoration phase.

Definition at line 23 of file IpAugRestoSystemSolver.hpp.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `Ipopt::AugRestoSystemSolver::AugRestoSystemSolver (AugSystemSolver & orig_aug_solver, bool skip_orig_aug_solver_init = true)`

Constructor.

Here, `orig_aug_solver` is the object for solving the original augmented system. The flag `skip_orig_aug_solver_init` indicates, if the initialization call (to `Initialize`) should be skipped; this flag will usually be true, so that the symbolic factorization of the main algorithm will be used.

6.8.2.2 `virtual Ipopt::AugRestoSystemSolver::~~AugRestoSystemSolver () [virtual]`

Default destructor.

6.8.2.3 `Ipopt::AugRestoSystemSolver::AugRestoSystemSolver () [private]`

Default Constructor.

6.8.2.4 `Ipopt::AugRestoSystemSolver::AugRestoSystemSolver (const AugRestoSystemSolver &) [private]`

Copy Constructor.

6.8.3 Member Function Documentation

6.8.3.1 `bool Ipopt::AugRestoSystemSolver::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AugSystemSolver](#).

6.8.3.2 `virtual ESymSolverStatus Ipopt::AugRestoSystemSolver::Solve (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix * J_c, const Vector * D_c, double delta_c, const Matrix * J_d, const Vector * D_d, double delta_d, const Vector & rhs_x, const Vector & rhs_s, const Vector & rhs_c, const Vector & rhs_d, Vector & sol_x, Vector & sol_s, Vector & sol_c, Vector & sol_d, bool check_NegEVals, Index numberOfNegEVals) [virtual]`

Translate the augmented system (in the full space of the restoration variables) into the smaller space of the original variables.

Reimplemented from [Ipopt::AugSystemSolver](#).

6.8.3.3 `virtual Index Ipopt::AugRestoSystemSolver::NumberOfNegEVals () const [inline], [virtual]`

Returns the number of negative eigenvalues from the original augmented system call.

Implements [Ipopt::AugSystemSolver](#).

Definition at line 76 of file `IpAugRestoSystemSolver.hpp`.

6.8.3.4 `virtual bool Ipopt::AugRestoSystemSolver::ProvidesInertia () const [inline], [virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::AugSystemSolver](#).

Definition at line 84 of file `IpAugRestoSystemSolver.hpp`.

6.8.3.5 `virtual bool Ipopt::AugRestoSystemSolver::IncreaseQuality () [inline],[virtual]`

Request to increase quality of solution for next solve.

Ask underlying linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::AugSystemSolver](#).

Definition at line 95 of file `IpAugRestoSystemSolver.hpp`.

6.8.3.6 `void Ipopt::AugRestoSystemSolver::operator= (const AugRestoSystemSolver &) [private]`

Overloaded Equals Operator.

6.8.3.7 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::Neg_Omega_c_plus_D_c (const SmartPtr<const Vector> & sigma_tilde_n_c_inv, const SmartPtr<const Vector> & sigma_tilde_p_c_inv, const Vector * D_c, const Vector & any_vec_in_c) [private]`

6.8.3.8 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::Neg_Omega_d_plus_D_d (const Matrix & Pd_L, const SmartPtr<const Vector> & sigma_tilde_n_d_inv, const Matrix & neg_Pd_U, const SmartPtr<const Vector> & sigma_tilde_p_d_inv, const Vector * D_d, const Vector & any_vec_in_d) [private]`

6.8.3.9 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::Sigma_tilde_n_c_inv (const SmartPtr<const Vector> & sigma_tilde_n_c, Number delta_x, const Vector & any_vec_in_n_c) [private]`

Sigma tilde is the sum of Sigma and delta_x times the identity.

6.8.3.10 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::Sigma_tilde_p_c_inv (const SmartPtr<const Vector> & sigma_tilde_p_c, Number delta_x, const Vector & any_vec_in_p_c) [private]`

6.8.3.11 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::Sigma_tilde_n_d_inv (const SmartPtr<const Vector> & sigma_tilde_n_d, Number delta_x, const Vector & any_vec_in_n_d) [private]`

6.8.3.12 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::Sigma_tilde_p_d_inv (const SmartPtr<const Vector> & sigma_tilde_p_d, Number delta_x, const Vector & any_vec_in_p_d) [private]`

6.8.3.13 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::D_x_plus_wr_d (const SmartPtr<const Vector> & CD_x0, Number factor, const Vector & wr_d) [private]`

6.8.3.14 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::Rhs_cR (const Vector & rhs_c, const SmartPtr<const Vector> & sigma_tilde_n_c_inv, const Vector & rhs_n_c, const SmartPtr<const Vector> & sigma_tilde_p_c_inv, const Vector & rhs_p_c) [private]`

6.8.3.15 `SmartPtr<const Vector> Ipopt::AugRestoSystemSolver::Rhs_dR (const Vector & rhs_d, const SmartPtr<const Vector> & sigma_tilde_n_d_inv, const Vector & rhs_n_d, const Matrix & pd_L, const SmartPtr<const Vector> & sigma_tilde_p_d_inv, const Vector & rhs_p_d, const Matrix & pd_U) [private]`

6.8.4 Member Data Documentation

6.8.4.1 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::neg_omega_c_plus_D_c_cache_ [private]`

Definition at line 121 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.2 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::neg_omega_d_plus_D_d_cache_`
`[private]`

Definition at line 122 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.3 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::sigma_tilde_n_c_inv_cache_`
`[private]`

Definition at line 123 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.4 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::sigma_tilde_p_c_inv_cache_`
`[private]`

Definition at line 124 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.5 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::sigma_tilde_n_d_inv_cache_`
`[private]`

Definition at line 125 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.6 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::sigma_tilde_p_d_inv_cache_`
`[private]`

Definition at line 126 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.7 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::d_x_plus_wr_d_cache_` `[private]`

Definition at line 127 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.8 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::rhs_cR_cache_` `[private]`

Definition at line 128 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.9 `CachedResults< SmartPtr<Vector> > Ipopt::AugRestoSystemSolver::rhs_dR_cache_` `[private]`

Definition at line 129 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.10 `SmartPtr<AugSystemSolver> Ipopt::AugRestoSystemSolver::orig_aug_solver_` `[private]`

Definition at line 191 of file `IpAugRestoSystemSolver.hpp`.

6.8.4.11 `bool Ipopt::AugRestoSystemSolver::skip_orig_aug_solver_init_` `[private]`

Definition at line 192 of file `IpAugRestoSystemSolver.hpp`.

The documentation for this class was generated from the following file:

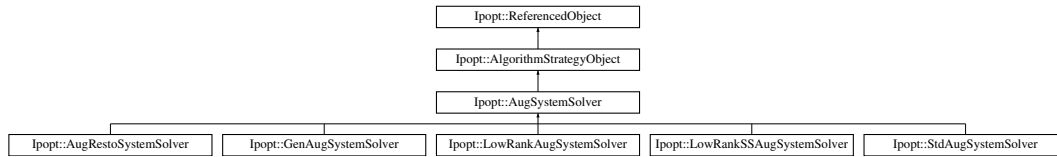
- [Algorithm/IpAugRestoSystemSolver.hpp](#)

6.9 Ipopt::AugSystemSolver Class Reference

Base class for Solver for the augmented system.

`#include <IpAugSystemSolver.hpp>`

Inheritance diagram for `Ipopt::AugSystemSolver`:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)
- virtual [ESymSolverStatus](#) [Solve](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) *J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) *J_d, const [Vector](#) *D_d, double delta_d, const [Vector](#) &rhs_x, const [Vector](#) &rhs_s, const [Vector](#) &rhs_c, const [Vector](#) &rhs_d, [Vector](#) &sol_x, [Vector](#) &sol_s, [Vector](#) &sol_c, [Vector](#) &sol_d, bool check_NegEVals, [Index](#) numberOfNegEVals)
Set up the augmented system and solve it for a given right hand side.
- virtual [ESymSolverStatus](#) [MultiSolve](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) *J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) *J_d, const [Vector](#) *D_d, double delta_d, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_xV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_sV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_cV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_dV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_xV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_sV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_cV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_dV, bool check_NegEVals, [Index](#) numberOfNegEVals)
Like Solve, but for multiple right hand sides.
- virtual [Index](#) [NumberOfNegEVals](#) () const =0
Number of negative eigenvalues detected during last solve.
- virtual bool [ProvidesInertia](#) () const =0
Query whether inertia is computed by linear solver.
- virtual bool [IncreaseQuality](#) ()=0
Request to increase quality of solution for next solve.

Constructors/Destructors

- [AugSystemSolver](#) ()
Default constructor.
- virtual [~AugSystemSolver](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [AugSystemSolver](#) (const [AugSystemSolver](#) &)
Copy Constructor.
- void [operator=](#) (const [AugSystemSolver](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.9.1 Detailed Description

Base class for Solver for the augmented system.

This is the base class for linear solvers that solve the augmented system, which is defined as

$$\begin{bmatrix} W + D_x + \delta_x I & 0 & J_c^T & J_d^T \\ 0 & D_s + \delta_s I & 0 & -I \\ J_c & 0 & D_c - \delta_c I & 0 \\ J_d & -I & 0 & D_d - \delta_d I \end{bmatrix} \begin{pmatrix} sol_x \\ sol_s \\ sol_c \\ sol_d \end{pmatrix} = \begin{pmatrix} rhs_x \\ rhs_s \\ rhs_c \\ rhs_d \end{pmatrix}$$

Since this system might be solved repeatedly for different right hand sides, it is desirable to step the factorization of a direct linear solver if possible.

Definition at line 37 of file IpAugSystemSolver.hpp.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Ipopt::AugSystemSolver::AugSystemSolver () [inline]

Default constructor.

Definition at line 43 of file IpAugSystemSolver.hpp.

6.9.2.2 virtual Ipopt::AugSystemSolver::~~AugSystemSolver () [inline],[virtual]

Default destructor.

Definition at line 46 of file IpAugSystemSolver.hpp.

6.9.2.3 Ipopt::AugSystemSolver::AugSystemSolver (const AugSystemSolver &) [private]

Copy Constructor.

6.9.3 Member Function Documentation

6.9.3.1 virtual bool Ipopt::AugSystemSolver::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::AugRestoSystemSolver](#), [Ipopt::LowRankSSAugSystemSolver](#), [Ipopt::StdAugSystemSolver](#), [Ipopt::LowRankAugSystemSolver](#), and [Ipopt::GenAugSystemSolver](#).

6.9.3.2 virtual ESymSolverStatus Ipopt::AugSystemSolver::Solve (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix * J_c, const Vector * D_c, double delta_c, const Matrix * J_d, const Vector * D_d, double delta_d, const Vector & rhs_x, const Vector & rhs_s, const Vector & rhs_c, const Vector & rhs_d, Vector & sol_x, Vector & sol_s, Vector & sol_c, Vector & sol_d, bool check_NegEvals, Index numberOfNegEvals) [inline],[virtual]

Set up the augmented system and solve it for a given right hand side.

If desired (i.e. if check_NegEvals is true), then the solution is only computed if the number of negative eigenvalues matches numberOfNegEvals.

The return value is the return value of the linear solver object.

Reimplemented in [Ipopt::AugRestoSystemSolver](#), [Ipopt::LowRankSSAugSystemSolver](#), and [Ipopt::LowRankAugSystemSolver](#).

Definition at line 61 of file `IpAugSystemSolver.hpp`.

```
6.9.3.3 virtual ESymSolverStatus Ipopt::AugSystemSolver::MultiSolve ( const SymMatrix * W, double W_factor, const
    Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix * J_c, const Vector * D_c, double
    delta_c, const Matrix * J_d, const Vector * D_d, double delta_d, std::vector< SmartPtr< const Vector > > &
    rhs_xV, std::vector< SmartPtr< const Vector > > & rhs_sV, std::vector< SmartPtr< const Vector > > & rhs_cV,
    std::vector< SmartPtr< const Vector > > & rhs_dV, std::vector< SmartPtr< Vector > > & sol_xV, std::vector<
    SmartPtr< Vector > > & sol_sV, std::vector< SmartPtr< Vector > > & sol_cV, std::vector< SmartPtr< Vector
    > > & sol_dV, bool check_NegEVals, Index numberOfNegEVals ) [inline],[virtual]
```

Like Solve, but for multiple right hand sides.

The inheriting class has to be overload at least one of Solve and MultiSolve.

Reimplemented in [Ipopt::StdAugSystemSolver](#), and [Ipopt::GenAugSystemSolver](#).

Definition at line 110 of file `IpAugSystemSolver.hpp`.

```
6.9.3.4 virtual Index Ipopt::AugSystemSolver::NumberOfNegEVals ( ) const [pure virtual]
```

Number of negative eigenvalues detected during last solve.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see ProvidesInertia).

Implemented in [Ipopt::LowRankSSAugSystemSolver](#), [Ipopt::StdAugSystemSolver](#), [Ipopt::AugRestoSystemSolver](#), [Ipopt::LowRankAugSystemSolver](#), and [Ipopt::GenAugSystemSolver](#).

```
6.9.3.5 virtual bool Ipopt::AugSystemSolver::ProvidesInertia ( ) const [pure virtual]
```

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implemented in [Ipopt::AugRestoSystemSolver](#), [Ipopt::LowRankSSAugSystemSolver](#), [Ipopt::StdAugSystemSolver](#), [Ipopt::LowRankAugSystemSolver](#), and [Ipopt::GenAugSystemSolver](#).

```
6.9.3.6 virtual bool Ipopt::AugSystemSolver::IncreaseQuality ( ) [pure virtual]
```

Request to increase quality of solution for next solve.

Ask underlying linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implemented in [Ipopt::AugRestoSystemSolver](#), [Ipopt::LowRankSSAugSystemSolver](#), [Ipopt::StdAugSystemSolver](#), [Ipopt::LowRankAugSystemSolver](#), and [Ipopt::GenAugSystemSolver](#).

```
6.9.3.7 void Ipopt::AugSystemSolver::operator= ( const AugSystemSolver & ) [private]
```

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

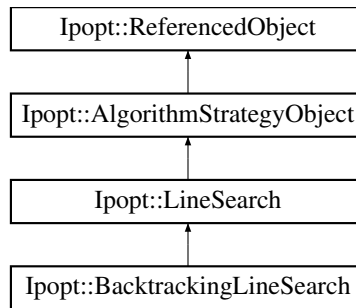
- [Algorithm/IpAugSystemSolver.hpp](#)

6.10 Ipopt::BacktrackingLineSearch Class Reference

General implementation of a backtracking line search.

```
#include <IpBacktrackingLineSearch.hpp>
```

Inheritance diagram for Ipopt::BacktrackingLineSearch:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
InitializeImpl - overloaded from [AlgorithmStrategyObject](#).
- virtual void [FindAcceptableTrialPoint](#) ()
Perform the line search.
- virtual void [Reset](#) ()
Reset the line search.
- virtual void [SetRigorousLineSearch](#) (bool rigorous)
Set flag indicating whether a very rigorous line search should be performed.
- virtual bool [CheckSkippedLineSearch](#) ()
Check if the line search procedure didn't accept a new iterate during the last call of [FindAcceptableTrialPoint\(\)](#).
- virtual bool [ActivateFallbackMechanism](#) ()
Activate fallback mechanism.

Constructors/Destructors

- [BacktrackingLineSearch](#) (const [SmartPtr](#)< [BacktrackingLSAcceptor](#) > &acceptor, const [SmartPtr](#)< [RestorationPhase](#) > &resto_phase, const [SmartPtr](#)< [ConvergenceCheck](#) > &conv_check)
Constructor.
- virtual [~BacktrackingLineSearch](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for [OptionsList](#).

Private Member Functions

- bool [DoBacktrackingLineSearch](#) (bool skip_first_trial_point, [Number](#) &alpha_primal, bool &corr_taken, bool &soc_taken, [Index](#) &n_steps, bool &evaluation_error, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Method performing the backtracking line search.
- void [StartWatchDog](#) ()
Method for starting the watch dog.
- void [StopWatchDog](#) ([SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Method for stopping the watch dog.
- bool [CheckAcceptabilityOfTrialPoint](#) ([Number](#) alpha_primal)
Method for checking if current trial point is acceptable.
- void [PerformDualStep](#) ([Number](#) alpha_primal, [Number](#) alpha_dual, [SmartPtr](#)< [IteratesVector](#) > &delta)
Method for setting the dual variables in the trial fields in IpData, given the search direction.
- bool [TrySoftRestoStep](#) ([SmartPtr](#)< [IteratesVector](#) > &actual_delta, bool &satisfies_original_criterion)
Try a step for the soft restoration phase and check if it is acceptable.
- bool [TrySecondOrderCorrection](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try a second order correction for the constraints.
- bool [TryCorrector](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try higher order corrector (for fast local convergence).
- void [PerformMagicStep](#) ()
Perform magic steps.
- bool [DetectTinyStep](#) ()
Detect if the search direction is too small.
- void [StoreAcceptablePoint](#) ()
Store current iterate as acceptable point.
- bool [RestoreAcceptablePoint](#) ()
Restore acceptable point into the current fields of IpData if found.
- bool [CurrentIsAcceptable](#) ()
Method for determining if the current iterate is acceptable (in the sense of the acceptable_tol options).

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [BacktrackingLineSearch](#) (const [BacktrackingLineSearch](#) &)
Copy Constructor.
- void [operator=](#) (const [BacktrackingLineSearch](#) &)
Overloaded Equals Operator.

Private Attributes

- bool [fallback_activated_](#)
Flag indicating whether the algorithm has asked to immediately switch to the fallback mechanism (restoration phase)
- bool [rigorous_](#)
Flag indicating whether the line search is to be performed robust (usually this is true, unless SetRigorousLineSearch is called with false).

- bool [skipped_line_search_](#)
Flag indicating whether no acceptable trial point was found during last line search.
- bool [in_soft_resto_phase_](#)
Flag indicating whether we are currently in the "soft" restoration phase mode, in which steps are accepted if they reduce the optimality error (see [soft_resto_pderror_reduction_factor](#))
- [Index soft_resto_counter_](#)
Counter for iteration performed in soft restoration phase in a row.
- [Index count_successive_shortened_steps_](#)
Counter for the number of successive iterations in which the full step was not accepted.
- bool [tiny_step_last_iteration_](#)
Flag indicating if a tiny step was detected in previous iteration.

Information related to watchdog procedure

- bool [in_watchdog_](#)
Flag indicating if the watchdog is active.
- [Index watchdog_shortened_iter_](#)
Counter for shortened iterations.
- [Index watchdog_trial_iter_](#)
Counter for watch dog iterations.
- [Number watchdog_alpha_primal_test_](#)
Step size for Armijo test in watch dog.
- [SmartPtr< const IteratesVector > watchdog_iterate_](#)
Watchdog reference iterate.
- [SmartPtr< const IteratesVector > watchdog_delta_](#)
Watchdog search direction at reference point.
- [Number last_mu_](#)
Barrier parameter value during last line search.

Storage for last iterate that satisfies the acceptable

level of optimality error.

- [SmartPtr< const IteratesVector > acceptable_iterate_](#)
- [Index acceptable_iteration_number_](#)

Strategy objective that are used

- [SmartPtr< BacktrackingLSAcceptor > acceptor_](#)
- [SmartPtr< RestorationPhase > resto_phase_](#)
- [SmartPtr< ConvergenceCheck > conv_check_](#)

Parameters for the filter algorithm. Names as in the paper

- enum [AlphaForYEnum](#) {
PRIMAL_ALPHA_FOR_Y=0, DUAL_ALPHA_FOR_Y, MIN_ALPHA_FOR_Y, MAX_ALPHA_FOR_Y,
FULL_STEP_FOR_Y, MIN_DUAL_INFEAS_ALPHA_FOR_Y, SAFE_MIN_DUAL_INFEAS_ALPHA_FOR_Y, P-
RIMAL_AND_FULL_ALPHA_FOR_Y,
DUAL_AND_FULL_ALPHA_FOR_Y, LSACCEPTOR_ALPHA_FOR_Y}
enumeration for the different alpha_for_y_ settings
- [Number alpha_red_factor_](#)
factor by which search direction is to be shortened if trial point is rejected.
- [AlphaForYEnum alpha_for_y_](#)

Flag indicating whether the dual step size is to be used for the equality constraint multipliers.

- [Number alpha_for_y_tol_](#)

Tolerance for primal step to switch to full equality constraint multiplier steps.

- [Number soft_resto_pderror_reduction_factor_](#)

Reduction factor for the restoration phase that accepts steps reducing the optimality error ("soft restoration phase").

- [Index max_soft_resto_iters_](#)

Maximal number of iterations that can be done in the soft iteration phase before the algorithm reverts to the regular restoration phase.

- [bool magic_steps_](#)

Flag indicating whether magic steps should be used.

- [bool accept_every_trial_step_](#)

Flag indicating whether the line search should always accept the full (fraction-to-the-boundary) step.

- [Index accept_after_max_steps_](#)

Maximal number of trial steps before we blindly accept trial point.

- [bool expect_infeasible_problem_](#)

Indicates whether problem can be expected to be infeasible.

- [Number expect_infeasible_problem_ctol_](#)

Tolerance on constraint violation for expect_infeasible_problem heuristic.

- [Number expect_infeasible_problem_ytol_](#)

Trigger tolerance on constraint multipliers.

- [Number tiny_step_tol_](#)

Tolerance for detecting tiny steps.

- [Number tiny_step_y_tol_](#)

Tolerance for y variables for the tiny step stopping heuristic.

- [Index watchdog_trial_iter_max_](#)

Number of watch dog trial steps.

- [Index watchdog_shortened_iter_trigger_](#)

Number of shortened iterations that trigger the watchdog.

- [bool start_with_resto_](#)

Indicates whether the algorithm should start directly with the restoration phase.

Additional Inherited Members

6.10.1 Detailed Description

General implementation of a backtracking line search.

This class can be used to perform the filter line search procedure or other procedures. The [BacktrackingLSAcceptor](#) is used to determine whether trial points are acceptable (e.g., based on a filter or other methods).

This backtracking line search knows of a restoration phase (which is called when the trial step size becomes too small or no search direction could be computed). It also has the notion of a "soft restoration phase," which uses the regular steps but decides on the acceptability based on other measures than the regular ones (e.g., reduction of the PD error instead of acceptability to a filter mechanism).

Definition at line 36 of file IpBacktrackingLineSearch.hpp.

6.10.2 Member Enumeration Documentation

6.10.2.1 enum Ipopt::BacktrackingLineSearch::AlphaForYEnum [private]

enumeration for the different alpha_for_y_ settings

Enumerator

```
PRIMAL_ALPHA_FOR_Y
DUAL_ALPHA_FOR_Y
MIN_ALPHA_FOR_Y
MAX_ALPHA_FOR_Y
FULL_STEP_FOR_Y
MIN_DUAL_INFEAS_ALPHA_FOR_Y
SAFE_MIN_DUAL_INFEAS_ALPHA_FOR_Y
PRIMAL_AND_FULL_ALPHA_FOR_Y
DUAL_AND_FULL_ALPHA_FOR_Y
LSACCEPTOR_ALPHA_FOR_Y
```

Definition at line 228 of file IpBacktrackingLineSearch.hpp.

6.10.3 Constructor & Destructor Documentation

6.10.3.1 Ipopt::BacktrackingLineSearch::BacktrackingLineSearch (const SmartPtr< BacktrackingLSAcceptor > & acceptor, const SmartPtr< RestorationPhase > & resto_phase, const SmartPtr< ConvergenceCheck > & conv_check)

Constructor.

The acceptor implements the acceptance test for the line search. The [ConvergenceCheck](#) object is used to determine whether the current iterate is acceptable (for example, the restoration phase is not started if the acceptability level has been reached). If conv_check is NULL, we assume that the current iterate is not acceptable (in the sense of the acceptable_tol option).

6.10.3.2 virtual Ipopt::BacktrackingLineSearch::~~BacktrackingLineSearch () [virtual]

Default destructor.

6.10.3.3 Ipopt::BacktrackingLineSearch::BacktrackingLineSearch (const BacktrackingLineSearch &) [private]

Copy Constructor.

6.10.4 Member Function Documentation

6.10.4.1 virtual bool Ipopt::BacktrackingLineSearch::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

InitializeImpl - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::AlgorithmStrategyObject](#).

6.10.4.2 `virtual void Ipopt::BacktrackingLineSearch::FindAcceptableTrialPoint () [virtual]`

Perform the line search.

It is assumed that the search direction is computed in the data object.

Implements [Ipopt::LineSearch](#).

6.10.4.3 `virtual void Ipopt::BacktrackingLineSearch::Reset () [virtual]`

Reset the line search.

This function should be called if all previous information should be discarded when the line search is performed the next time. For example, this method should be called if the barrier parameter is changed.

Implements [Ipopt::LineSearch](#).

6.10.4.4 `virtual void Ipopt::BacktrackingLineSearch::SetRigorousLineSearch (bool rigorous) [inline],[virtual]`

Set flag indicating whether a very rigorous line search should be performed.

If this flag is set to true, the line search algorithm might decide to abort the line search and not to accept a new iterate. If the line search decided not to accept a new iterate, the return value of [CheckSkippedLineSearch\(\)](#) is true at the next call. For example, in the non-monotone barrier parameter update procedure, the filter algorithm should not switch to the restoration phase in the free mode; instead, the algorithm should switch to the fixed mode.

Implements [Ipopt::LineSearch](#).

Definition at line 85 of file `IpBacktrackingLineSearch.hpp`.

6.10.4.5 `virtual bool Ipopt::BacktrackingLineSearch::CheckSkippedLineSearch () [inline],[virtual]`

Check if the line search procedure didn't accept a new iterate during the last call of [FindAcceptableTrialPoint\(\)](#).

Implements [Ipopt::LineSearch](#).

Definition at line 94 of file `IpBacktrackingLineSearch.hpp`.

6.10.4.6 `virtual bool Ipopt::BacktrackingLineSearch::ActivateFallbackMechanism () [virtual]`

Activate fallback mechanism.

Return false, if that is not possible.

Implements [Ipopt::LineSearch](#).

6.10.4.7 `static void Ipopt::BacktrackingLineSearch::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for [OptionsList](#).

6.10.4.8 `void Ipopt::BacktrackingLineSearch::operator= (const BacktrackingLineSearch &) [private]`

Overloaded Equals Operator.

6.10.4.9 `bool Ipopt::BacktrackingLineSearch::DoBacktrackingLineSearch (bool skip_first_trial_point, Number & alpha_primal, bool & corr_taken, bool & soc_taken, Index & n_steps, bool & evaluation_error, SmartPtr< IteratesVector > & actual_delta) [private]`

Method performing the backtracking line search.

The return value indicates if the step acceptance criteria are met. If the watchdog is active, only one trial step is

performed (and the trial values are set accordingly).

6.10.4.10 `void Ipopt::BacktrackingLineSearch::StartWatchDog () [private]`

Method for starting the watch dog.

Set all appropriate fields accordingly

6.10.4.11 `void Ipopt::BacktrackingLineSearch::StopWatchDog (SmartPtr< IteratesVector > & actual_delta) [private]`

Method for stopping the watch dog.

Set all appropriate fields accordingly.

6.10.4.12 `bool Ipopt::BacktrackingLineSearch::CheckAcceptabilityOfTrialPoint (Number alpha_primal) [private]`

Method for checking if current trial point is acceptable.

It is assumed that the delta information in ip_data is the search direction used in criteria. The primal trial point has to be set before the call.

6.10.4.13 `void Ipopt::BacktrackingLineSearch::PerformDualStep (Number alpha_primal, Number alpha_dual, SmartPtr< IteratesVector > & delta) [private]`

Method for setting the dual variables in the trial fields in IpData, given the search direction.

The step size for the bound multipliers is alpha_dual (the fraction-to-the-boundary step size), and the step size for the equality constraint multipliers depends on the choice of alpha_for_y.

6.10.4.14 `bool Ipopt::BacktrackingLineSearch::TrySoftRestoStep (SmartPtr< IteratesVector > & actual_delta, bool & satisfies_original_criterion) [private]`

Try a step for the soft restoration phase and check if it is acceptable.

The step size is identical for all variables. A point is accepted if it is acceptable for the original acceptability criterion (in which case satisfies_original_criterion = true on return), or if the primal-dual system error was decrease by at least the factor soft_resto_pderror_reduction_factor_. The return value is true, if the trial point was acceptable for the soft restoration phase.

6.10.4.15 `bool Ipopt::BacktrackingLineSearch::TrySecondOrderCorrection (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta) [private]`

Try a second order correction for the constraints.

If the first trial step (with incoming alpha_primal) has been reject, this tries up to max_soc_ second order corrections for the constraints. Here, alpha_primal_test is the step size that has to be used in the filter acceptance tests. On output actual_delta... has been set to the steps including the second order correction if it has been accepted, otherwise it is unchanged. If the SOC step has been accepted, alpha_primal has the fraction-to-the-boundary value for the SOC step on output. The return value is true, if an SOC step has been accepted.

6.10.4.16 `bool Ipopt::BacktrackingLineSearch::TryCorrector (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta) [private]`

Try higher order corrector (for fast local convergence).

In contrast to a second order correction step, which tries to make an unacceptable point acceptable by improving constraint violation, this corrector step is tried even if the regular primal-dual step is acceptable.

6.10.4.17 void Ipopt::BacktrackingLineSearch::PerformMagicStep () [private]

Perform magic steps.

Take the current values of the slacks in trial and replace them by better ones that lead to smaller values of the barrier function and less constraint violation.

6.10.4.18 bool Ipopt::BacktrackingLineSearch::DetectTinyStep () [private]

Detect if the search direction is too small.

This should be true if the search direction is so small that it makes numerically no difference.

6.10.4.19 void Ipopt::BacktrackingLineSearch::StoreAcceptablePoint () [private]

Store current iterate as acceptable point.

6.10.4.20 bool Ipopt::BacktrackingLineSearch::RestoreAcceptablePoint () [private]

Restore acceptable point into the current fields of IpData if found.

Returns true if such a point is available.

6.10.4.21 bool Ipopt::BacktrackingLineSearch::CurrentIsAcceptable () [private]

Method for determining if the current iterate is acceptable (in the sense of the `acceptable_tol` options).

This is a wrapper for same method from [ConvergenceCheck](#), but returns false, if no [ConvergenceCheck](#) object is provided.

6.10.5 Member Data Documentation**6.10.5.1 Number Ipopt::BacktrackingLineSearch::alpha_red_factor_ [private]**

factor by which search direction is to be shortened if trial point is rejected.

Definition at line 225 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.2 AlphaForYEnum Ipopt::BacktrackingLineSearch::alpha_for_y_ [private]

Flag indicating whether the dual step size is to be used for the equality constraint multipliers.

If 0, the primal step size is used, if 1 the dual step size, and if 2, the minimum of both.

Definition at line 245 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.3 Number Ipopt::BacktrackingLineSearch::alpha_for_y_tol_ [private]

Tolerance for primal step to switch to full equality constraint multiplier steps.

Definition at line 249 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.4 Number Ipopt::BacktrackingLineSearch::soft_resto_pderror_reduction_factor_ [private]

Reduction factor for the restoration phase that accepts steps reducing the optimality error ("soft restoration phase").

If 0., then this restoration phase is not enabled.

Definition at line 254 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.5 Index `Ipopt::BacktrackingLineSearch::max_soft_resto_iters_` `[private]`

Maximal number of iterations that can be done in the soft iteration phase before the algorithm reverts to the regular restoration phase.

Definition at line 258 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.6 bool `Ipopt::BacktrackingLineSearch::magic_steps_` `[private]`

Flag indicating whether magic steps should be used.

Definition at line 261 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.7 bool `Ipopt::BacktrackingLineSearch::accept_every_trial_step_` `[private]`

Flag indicating whether the line search should always accept the full (fraction-to-the-boundary) step.

Definition at line 264 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.8 Index `Ipopt::BacktrackingLineSearch::accept_after_max_steps_` `[private]`

Maximal number of trial steps before we blindly accept trial point.

If set to value other than -1, we accept a trial point even if it is not satisfying acceptance criteria.

Definition at line 268 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.9 bool `Ipopt::BacktrackingLineSearch::expect_infeasible_problem_` `[private]`

Indicates whether problem can be expected to be infeasible.

This will trigger requesting a tighter reduction in infeasibility the first time the restoration phase is called.

Definition at line 273 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.10 Number `Ipopt::BacktrackingLineSearch::expect_infeasible_problem_ctol_` `[private]`

Tolerance on constraint violation for `expect_infeasible_problem` heuristic.

If the constraint violation becomes that than this value, the heuristic is disabled for the rest of the optimization run.

Definition at line 278 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.11 Number `Ipopt::BacktrackingLineSearch::expect_infeasible_problem_ytol_` `[private]`

Trigger tolerance on constraint multipliers.

If `expect_infeasible_problem` is chosen, and the multipliers become larger in max-norm than this value, the restoration phase is triggered.

Definition at line 283 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.12 Number `Ipopt::BacktrackingLineSearch::tiny_step_tol_` `[private]`

Tolerance for detecting tiny steps.

Definition at line 286 of file `IpBacktrackingLineSearch.hpp`.

6.10.5.13 Number `Ipopt::BacktrackingLineSearch::tiny_step_y_tol_` `[private]`

Tolerance for y variables for the tiny step stopping heuristic.

If repeatedly a tiny step is detected and the step in the `y_c` and `y_d` variables is less than this threshold, we algorithm

will stop.

Definition at line 292 of file IpBacktrackingLineSearch.hpp.

6.10.5.14 Index Ipopt::BacktrackingLineSearch::watchdog_trial_iter_max_ [private]

Number of watch dog trial steps.

Definition at line 295 of file IpBacktrackingLineSearch.hpp.

6.10.5.15 Index Ipopt::BacktrackingLineSearch::watchdog_shortened_iter_trigger_ [private]

Number of shortened iterations that trigger the watchdog.

Definition at line 297 of file IpBacktrackingLineSearch.hpp.

6.10.5.16 bool Ipopt::BacktrackingLineSearch::start_with_resto_ [private]

Indicates whether the algorithm should start directly with the restoratin phase.

Definition at line 301 of file IpBacktrackingLineSearch.hpp.

6.10.5.17 bool Ipopt::BacktrackingLineSearch::in_watchdog_ [private]

Flag indicating if the watchdog is active.

Definition at line 307 of file IpBacktrackingLineSearch.hpp.

6.10.5.18 Index Ipopt::BacktrackingLineSearch::watchdog_shortened_iter_ [private]

Counter for shortened iterations.

Definition at line 309 of file IpBacktrackingLineSearch.hpp.

6.10.5.19 Index Ipopt::BacktrackingLineSearch::watchdog_trial_iter_ [private]

Counter for watch dog iterations.

Definition at line 311 of file IpBacktrackingLineSearch.hpp.

6.10.5.20 Number Ipopt::BacktrackingLineSearch::watchdog_alpha_primal_test_ [private]

Step size for Armijo test in watch dog.

Definition at line 313 of file IpBacktrackingLineSearch.hpp.

6.10.5.21 SmartPtr<const IteratesVector> Ipopt::BacktrackingLineSearch::watchdog_iterate_ [private]

Watchdog reference iterate.

Definition at line 315 of file IpBacktrackingLineSearch.hpp.

6.10.5.22 SmartPtr<const IteratesVector> Ipopt::BacktrackingLineSearch::watchdog_delta_ [private]

Watchdog search direction at reference point.

Definition at line 317 of file IpBacktrackingLineSearch.hpp.

6.10.5.23 Number Ipopt::BacktrackingLineSearch::last_mu_ [private]

Barrier parameter value during last line search.

Definition at line 319 of file IpBacktrackingLineSearch.hpp.

6.10.5.24 **SmartPtr<const IteratesVector> Ipopt::BacktrackingLineSearch::acceptable_iterate_** [private]

Definition at line 325 of file IpBacktrackingLineSearch.hpp.

6.10.5.25 **Index Ipopt::BacktrackingLineSearch::acceptable_iteration_number_** [private]

Definition at line 326 of file IpBacktrackingLineSearch.hpp.

6.10.5.26 **bool Ipopt::BacktrackingLineSearch::fallback_activated_** [private]

Flag indicating whether the algorithm has asked to immediately switch to the fallback mechanism (restoration phase)

Definition at line 331 of file IpBacktrackingLineSearch.hpp.

6.10.5.27 **bool Ipopt::BacktrackingLineSearch::rigorous_** [private]

Flag indicating whether the line search is to be performed robust (usually this is true, unless SetRigorousLineSearch is called with false).

Definition at line 337 of file IpBacktrackingLineSearch.hpp.

6.10.5.28 **bool Ipopt::BacktrackingLineSearch::skipped_line_search_** [private]

Flag indicating whether no acceptable trial point was found during last line search.

Definition at line 341 of file IpBacktrackingLineSearch.hpp.

6.10.5.29 **bool Ipopt::BacktrackingLineSearch::in_soft_resto_phase_** [private]

Flag indicating whether we are currently in the "soft" restoration phase mode, in which steps are accepted if they reduce the optimality error (see soft_resto_pderror_reduction_factor)

Definition at line 347 of file IpBacktrackingLineSearch.hpp.

6.10.5.30 **Index Ipopt::BacktrackingLineSearch::soft_resto_counter_** [private]

Counter for iteration performed in soft restoration phase in a row.

Definition at line 351 of file IpBacktrackingLineSearch.hpp.

6.10.5.31 **Index Ipopt::BacktrackingLineSearch::count_successive_shortened_steps_** [private]

Counter for the number of successive iterations in which the full step was not accepted.

Definition at line 355 of file IpBacktrackingLineSearch.hpp.

6.10.5.32 **bool Ipopt::BacktrackingLineSearch::tiny_step_last_iteration_** [private]

Flag indicating if a tiny step was detected in previous iteration.

Definition at line 359 of file IpBacktrackingLineSearch.hpp.

6.10.5.33 **SmartPtr<BacktrackingLSAcceptor> Ipopt::BacktrackingLineSearch::acceptor_** [private]

Definition at line 363 of file IpBacktrackingLineSearch.hpp.

6.10.5.34 **SmartPtr<RestorationPhase> Ipopt::BacktrackingLineSearch::resto_phase_** [private]

Definition at line 364 of file IpBacktrackingLineSearch.hpp.

6.10.5.35 SmartPtr<ConvergenceCheck> Ipopt::BacktrackingLineSearch::conv_check_ [private]

Definition at line 365 of file IpBacktrackingLineSearch.hpp.

The documentation for this class was generated from the following file:

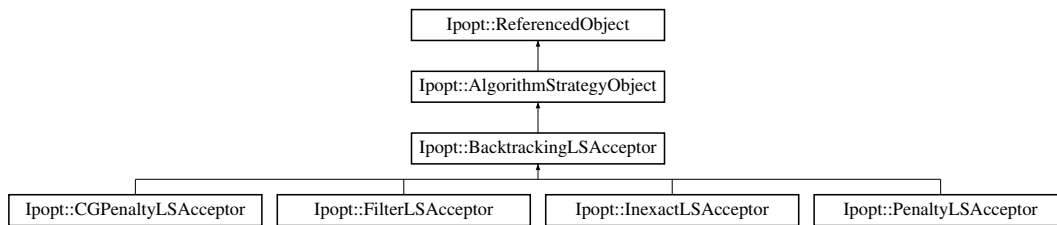
- [Algorithm/IpBacktrackingLineSearch.hpp](#)

6.11 Ipopt::BacktrackingLSAcceptor Class Reference

Base class for backtracking line search acceptors.

```
#include <IpBacktrackingLSAcceptor.hpp>
```

Inheritance diagram for Ipopt::BacktrackingLSAcceptor:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
InitializeImpl - overloaded from [AlgorithmStrategyObject](#).
- virtual void [Reset](#) ()=0
Reset the acceptor.
- virtual void [InitThisLineSearch](#) (bool in_watchdog)=0
Initialization for the next line search.
- virtual void [PrepareRestoPhaseStart](#) ()=0
Method that is called before the restoration phase is called.
- virtual [Number](#) [CalculateAlphaMin](#) ()=0
Method returning the lower bound on the trial step sizes.
- virtual bool [CheckAcceptabilityOfTrialPoint](#) ([Number](#) alpha_primal)=0
Method for checking if current trial point is acceptable.
- virtual bool [TrySecondOrderCorrection](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)=0
Try a second order correction for the constraints.
- virtual bool [TryCorrector](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)=0
Try higher order corrector (for fast local convergence).
- virtual char [UpdateForNextIteration](#) ([Number](#) alpha_primal_test)=0
Method for ending the current line search.
- virtual void [StartWatchDog](#) ()=0
Method for setting internal data if the watchdog procedure is started.
- virtual void [StopWatchDog](#) ()=0
Method for setting internal data if the watchdog procedure is stopped.

- virtual bool `RestoredIterate ()`
Method for telling the `BacktrackingLineSearch` object that a previous iterate has been restored.
- virtual bool `NeverRestorationPhase ()`
Method called by `BacktrackingLineSearch` object to determine whether the restoration phase should never be called.
- virtual bool `DoFallback ()`
Method for doing a fallback approach in case no search direction could be computed.
- virtual `Number ComputeAlphaForY (Number alpha_primal, Number alpha_dual, SmartPtr< IteratesVector > &delta)`
Method for computing the step for the constraint multipliers in the line search acceptor method.
- virtual bool `HasComputeAlphaForY () const`
Method returning true if `ComputeAlphaForY` is implemented for this acceptor.

Constructors/Destructors

- `BacktrackingLSAcceptor ()`
Constructor.
- virtual `~BacktrackingLSAcceptor ()`
Default destructor.

Static Public Member Functions

- static void `RegisterOptions (SmartPtr< RegisteredOptions > roptions)`
Methods for `OptionsList`.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `BacktrackingLSAcceptor (const BacktrackingLSAcceptor &)`
Copy Constructor.
- void `operator= (const BacktrackingLSAcceptor &)`
Overloaded Equals Operator.

Additional Inherited Members

6.11.1 Detailed Description

Base class for backtracking line search acceptors.

Definition at line 21 of file `IpBacktrackingLSAcceptor.hpp`.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 `Ipopt::BacktrackingLSAcceptor::BacktrackingLSAcceptor () [inline]`

Constructor.

Definition at line 27 of file `IpBacktrackingLSAcceptor.hpp`.

6.11.2.2 `virtual Ipopt::BacktrackingLSAcceptor::~BacktrackingLSAcceptor () [inline],[virtual]`

Default destructor.

Definition at line 31 of file IpBacktrackingLSAcceptor.hpp.

6.11.2.3 `Ipopt::BacktrackingLSAcceptor::BacktrackingLSAcceptor (const BacktrackingLSAcceptor &) [private]`

Copy Constructor.

6.11.3 Member Function Documentation

6.11.3.1 `virtual bool Ipopt::BacktrackingLSAcceptor::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]`

InitializeImpl - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::FilterLSAcceptor](#), [Ipopt::PenaltyLSAcceptor](#), [Ipopt::CGPenaltyLSAcceptor](#), and [Ipopt::InexactLSAcceptor](#).

6.11.3.2 `virtual void Ipopt::BacktrackingLSAcceptor::Reset () [pure virtual]`

Reset the acceptor.

This function should be called if all previous information should be discarded when the line search is performed the next time. For example, this method should be called if the barrier parameter is changed.

Implemented in [Ipopt::FilterLSAcceptor](#), [Ipopt::PenaltyLSAcceptor](#), [Ipopt::CGPenaltyLSAcceptor](#), and [Ipopt::InexactLSAcceptor](#).

6.11.3.3 `virtual void Ipopt::BacktrackingLSAcceptor::InitThisLineSearch (bool in_watchdog) [pure virtual]`

Initialization for the next line search.

The flag `in_watchdog` indicates if we are currently in an active watchdog procedure.

Implemented in [Ipopt::FilterLSAcceptor](#), [Ipopt::PenaltyLSAcceptor](#), [Ipopt::CGPenaltyLSAcceptor](#), and [Ipopt::InexactLSAcceptor](#).

6.11.3.4 `virtual void Ipopt::BacktrackingLSAcceptor::PrepareRestoPhaseStart () [pure virtual]`

Method that is called before the restoration phase is called.

Here, we can set up things that are required in the termination test for the restoration phase, such as augmenting a filter.

Implemented in [Ipopt::FilterLSAcceptor](#), [Ipopt::PenaltyLSAcceptor](#), [Ipopt::CGPenaltyLSAcceptor](#), and [Ipopt::InexactLSAcceptor](#).

6.11.3.5 `virtual Number Ipopt::BacktrackingLSAcceptor::CalculateAlphaMin () [pure virtual]`

Method returning the lower bound on the trial step sizes.

If the backtracking procedure encounters a trial step size below this value after the first trial set, it switches to the (soft) restoration phase.

Implemented in [Ipopt::CGPenaltyLSAcceptor](#), [Ipopt::FilterLSAcceptor](#), [Ipopt::PenaltyLSAcceptor](#), and [Ipopt::InexactLSAcceptor](#).

6.11.3.6 `virtual bool lpopt::BacktrackingLSAcceptor::CheckAcceptabilityOfTrialPoint (Number alpha_primal)` [pure virtual]

Method for checking if current trial point is acceptable.

It is assumed that the delta information in `ip_data` is the search direction used in criteria. The primal trial point has to be set before the call. `alpha_primal` is the step size which is to be used for the test; if it is zero, then this method is called during the soft restoration phase.

Implemented in [lpopt::CGPenaltyLSAcceptor](#), [lpopt::FilterLSAcceptor](#), [lpopt::PenaltyLSAcceptor](#), and [lpopt::InexactLSAcceptor](#).

6.11.3.7 `virtual bool lpopt::BacktrackingLSAcceptor::TrySecondOrderCorrection (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta)` [pure virtual]

Try a second order correction for the constraints.

If the first trial step (with incoming `alpha_primal`) has been reject, this tries second order corrections, e.g., for the constraints. Here, `alpha_primal_test` is the step size that has to be used in the filter acceptance tests. On output `actual_delta` has been set to the step including the second order correction if it has been accepted, otherwise it is unchanged. If the SOC step has been accepted, `alpha_primal` has the fraction-to-the-boundary value for the SOC step on output. The return value is true, if a SOC step has been accepted.

Implemented in [lpopt::CGPenaltyLSAcceptor](#), [lpopt::FilterLSAcceptor](#), [lpopt::PenaltyLSAcceptor](#), and [lpopt::InexactLSAcceptor](#).

6.11.3.8 `virtual bool lpopt::BacktrackingLSAcceptor::TryCorrector (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta)` [pure virtual]

Try higher order corrector (for fast local convergence).

In contrast to a second order correction step, which tries to make an unacceptable point acceptable by improving constraint violation, this corrector step is tried even if the regular primal-dual step is acceptable.

Implemented in [lpopt::CGPenaltyLSAcceptor](#), [lpopt::FilterLSAcceptor](#), [lpopt::PenaltyLSAcceptor](#), and [lpopt::InexactLSAcceptor](#).

6.11.3.9 `virtual char lpopt::BacktrackingLSAcceptor::UpdateForNextIteration (Number alpha_primal_test)` [pure virtual]

Method for ending the current line search.

When it is called, the internal data should be updates, e.g., the filter might be augmented. `alpha_primal_test` is the value of alpha that has been used for in the acceptance test earlier. Return value is a character for the `info_alpha_primal_char` field in `IpData`.

Implemented in [lpopt::CGPenaltyLSAcceptor](#), [lpopt::FilterLSAcceptor](#), [lpopt::PenaltyLSAcceptor](#), and [lpopt::InexactLSAcceptor](#).

6.11.3.10 `virtual void lpopt::BacktrackingLSAcceptor::StartWatchDog ()` [pure virtual]

Method for setting internal data if the watchdog procedure is started.

Implemented in [lpopt::CGPenaltyLSAcceptor](#), [lpopt::FilterLSAcceptor](#), [lpopt::PenaltyLSAcceptor](#), and [lpopt::InexactLSAcceptor](#).

6.11.3.11 `virtual void lpopt::BacktrackingLSAcceptor::StopWatchDog ()` [pure virtual]

Method for setting internal data if the watchdog procedure is stopped.

Implemented in [lpopt::CGPenaltyLSAcceptor](#), [lpopt::FilterLSAcceptor](#), [lpopt::PenaltyLSAcceptor](#), and [lpopt::InexactLSAcceptor](#).

[Acceptor](#).

6.11.3.12 `virtual bool Ipopt::BacktrackingLSAcceptor::RestoredIterate () [inline],[virtual]`

Method for telling the [BacktrackingLineSearch](#) object that a previous iterate has been restored.

Reimplemented in [Ipopt::CGPenaltyLSAcceptor](#).

Definition at line 115 of file `IpBacktrackingLSAcceptor.hpp`.

6.11.3.13 `virtual bool Ipopt::BacktrackingLSAcceptor::NeverRestorationPhase () [inline],[virtual]`

Method called by [BacktrackingLineSearch](#) object to determine whether the restoration phase should never be called.

Reimplemented in [Ipopt::CGPenaltyLSAcceptor](#).

Definition at line 122 of file `IpBacktrackingLSAcceptor.hpp`.

6.11.3.14 `virtual bool Ipopt::BacktrackingLSAcceptor::DoFallback () [inline],[virtual]`

Method for doing a fallback approach in case no search direction could be computed.

If no such fall back option is available, return false. If possible, the new point is assumed to be in the trial fields of `IpData` now.

Reimplemented in [Ipopt::CGPenaltyLSAcceptor](#).

Definition at line 131 of file `IpBacktrackingLSAcceptor.hpp`.

6.11.3.15 `virtual Number Ipopt::BacktrackingLSAcceptor::ComputeAlphaForY (Number alpha_primal, Number alpha_dual, SmartPtr< IteratesVector > & delta) [inline],[virtual]`

Method for computing the step for the constraint multipliers in the line search acceptor method.

This is activated with choosing the option `alpha_for_y=acceptor`

Reimplemented in [Ipopt::InexactLSAcceptor](#).

Definition at line 139 of file `IpBacktrackingLSAcceptor.hpp`.

6.11.3.16 `virtual bool Ipopt::BacktrackingLSAcceptor::HasComputeAlphaForY () const [inline],[virtual]`

Method returning true if `ComputeAlphaForY` is implemented for this acceptor.

Reimplemented in [Ipopt::InexactLSAcceptor](#).

Definition at line 150 of file `IpBacktrackingLSAcceptor.hpp`.

6.11.3.17 `static void Ipopt::BacktrackingLSAcceptor::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for [OptionsList](#).

6.11.3.18 `void Ipopt::BacktrackingLSAcceptor::operator= (const BacktrackingLSAcceptor &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

- [Algorithm/IpBacktrackingLSAcceptor.hpp](#)

6.12 Ipopt::CachedResults< T > Class Template Reference

Cache Priority Enum.

```
#include <IpCachedResults.hpp>
```

Public Member Functions

- bool [InvalidateResult](#) (const std::vector< const [TaggedObject](#) * > &dependents, const std::vector< [Number](#) > &scalar_dependents)
Invalidates the result for given dependencies.
- void [Clear](#) ()
Invalidates all cached results.
- void [Clear](#) (Int max_cache_size)
Invalidate all cached results and changes max_cache_size.

Constructors and Destructors.

- [CachedResults](#) (Int max_cache_size)
Constructor, where max_cache_size is the maximal number of results that should be cached.
- virtual [~CachedResults](#) ()
Destructor.

Generic methods for adding and retrieving cached results.

- void [AddCachedResult](#) (const T &result, const std::vector< const [TaggedObject](#) * > &dependents, const std::vector< [Number](#) > &scalar_dependents)
Generic method for adding a result to the cache, given a std::vector of TaggedObjects and a std::vector of Numbers.
- bool [GetCachedResult](#) (T &retResult, const std::vector< const [TaggedObject](#) * > &dependents, const std::vector< [Number](#) > &scalar_dependents) const
Generic method for retrieving a cached results, given the dependencies as a std::vector of TaggedObjects and a std::vector of Numbers.
- void [AddCachedResult](#) (const T &result, const std::vector< const [TaggedObject](#) * > &dependents)
Method for adding a result, providing only a std::vector of TaggedObjects.
- bool [GetCachedResult](#) (T &retResult, const std::vector< const [TaggedObject](#) * > &dependents) const
Method for retrieving a cached result, providing only a std::vector of TaggedObjects.

Pointer-based methods for adding and retrieving cached

results, providing dependencies explicitly.

- void [AddCachedResult1Dep](#) (const T &result, const [TaggedObject](#) *dependent1)
Method for adding a result to the cache, proving one dependency as a TaggedObject explicitly.
- bool [GetCachedResult1Dep](#) (T &retResult, const [TaggedObject](#) *dependent1)
Method for retrieving a cached result, proving one dependency as a TaggedObject explicitly.
- void [AddCachedResult2Dep](#) (const T &result, const [TaggedObject](#) *dependent1, const [TaggedObject](#) *dependent2)
Method for adding a result to the cache, proving two dependencies as a TaggedObject explicitly.
- bool [GetCachedResult2Dep](#) (T &retResult, const [TaggedObject](#) *dependent1, const [TaggedObject](#) *dependent2)
Method for retrieving a cached result, proving two dependencies as a TaggedObject explicitly.
- void [AddCachedResult3Dep](#) (const T &result, const [TaggedObject](#) *dependent1, const [TaggedObject](#) *dependent2, const [TaggedObject](#) *dependent3)
Method for adding a result to the cache, proving three dependencies as a TaggedObject explicitly.

- bool [GetCachedResult3Dep](#) (T &retResult, const [TaggedObject](#) *dependent1, const [TaggedObject](#) *dependent2, const [TaggedObject](#) *dependent3)

Method for retrieving a cached result, proving three dependencies as a [TaggedObject](#) explicitly.

Pointer-free version of the Add and Get methods

- bool [GetCachedResult1Dep](#) (T &retResult, const [TaggedObject](#) &dependent1)
- bool [GetCachedResult2Dep](#) (T &retResult, const [TaggedObject](#) &dependent1, const [TaggedObject](#) &dependent2)
- bool [GetCachedResult3Dep](#) (T &retResult, const [TaggedObject](#) &dependent1, const [TaggedObject](#) &dependent2, const [TaggedObject](#) &dependent3)
- void [AddCachedResult1Dep](#) (const T &result, const [TaggedObject](#) &dependent1)
- void [AddCachedResult2Dep](#) (const T &result, const [TaggedObject](#) &dependent1, const [TaggedObject](#) &dependent2)
- void [AddCachedResult3Dep](#) (const T &result, const [TaggedObject](#) &dependent1, const [TaggedObject](#) &dependent2, const [TaggedObject](#) &dependent3)

Private Member Functions

- void [CleanupInvalidatedResults](#) () const
internal method for removing stale DependentResults from the list.
- void [DebugPrintCachedResults](#) () const
Print list of currently cached results.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CachedResults](#) ()
Default Constructor.
- [CachedResults](#) (const [CachedResults](#) &)
Copy Constructor.
- void [operator=](#) (const [CachedResults](#) &)
Overloaded Equals Operator.

Private Attributes

- [Int](#) [max_cache_size_](#)
maximum number of cached results
- [std::list](#)< [DependentResult](#)< T > * > * [cached_results_](#)
list of currently cached results.

6.12.1 Detailed Description

```
template<class T>class Ipopt::CachedResults< T >
```

Cache Priority Enum.

Templated class for Cached Results. This class stores up to a given number of "results", entities that are stored here together with identifiers, that can be used to later retrieve the information again.

Typically, `T` is a [SmartPtr](#) for some calculated quantity that should be stored (such as a [Vector](#)). The identifiers (or dependencies) are a (possibly varying) number of Tags from [TaggedObjects](#), and a number of Numbers. Results are added to the cache using the `AddCachedResults` methods, and the can be retrieved with the `GetCachedResults` methods. The second set of methods checks whether a result has been cached for the given identifiers. If a corresponding result is found, a copy of it is returned and the method evaluates to true, otherwise it evaluates to false.

Note that cached results can become "stale", namely when a [TaggedObject](#) that is used to identify this `CachedResult` is changed. When this happens, the cached result can never be asked for again, so that there is no point in storing it any longer. For this purpose, a cached result, which is stored as a [DependentResult](#), inherits off an [Observer](#). This [Observer](#) retrieves notification whenever a [TaggedObject](#) dependency has changed. Stale results are later removed from the cache.

Definition at line 70 of file `IpCachedResults.hpp`.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `template<class T> Ipopt::CachedResults< T >::CachedResults (Int max_cache_size)`

Constructor, where `max_cache_size` is the maximal number of results that should be cached.

If `max_cache_size` is negative, we allow an infinite amount of cache.

Definition at line 476 of file `IpCachedResults.hpp`.

6.12.2.2 `template<class T> Ipopt::CachedResults< T >::~~CachedResults () [virtual]`

Destructor.

Definition at line 488 of file `IpCachedResults.hpp`.

6.12.2.3 `template<class T> Ipopt::CachedResults< T >::CachedResults () [private]`

Default Constructor.

6.12.2.4 `template<class T> Ipopt::CachedResults< T >::CachedResults (const CachedResults< T > &) [private]`

Copy Constructor.

6.12.3 Member Function Documentation

6.12.3.1 `template<class T> void Ipopt::CachedResults< T >::AddCachedResult (const T & result, const std::vector< const TaggedObject * > & dependents, const std::vector< Number > & scalar_dependents)`

Generic method for adding a result to the cache, given a `std::vector` of `TaggedObjects` and a `std::vector` of Numbers.

Definition at line 513 of file `IpCachedResults.hpp`.

6.12.3.2 `template<class T> bool Ipopt::CachedResults< T >::GetCachedResult (T & retResult, const std::vector< const TaggedObject * > & dependents, const std::vector< Number > & scalar_dependents) const`

Generic method for retrieving a cached results, given the dependencies as a `std::vector` of `TaggedObjects` and a `std::vector` of Numbers.

Definition at line 555 of file `IpCachedResults.hpp`.

6.12.3.3 `template<class T> void Ipopt::CachedResults< T >::AddCachedResult (const T & result, const std::vector< const TaggedObject * > & dependents)`

Method for adding a result, providing only a std::vector of TaggedObjects.

Definition at line 547 of file IpCachedResults.hpp.

6.12.3.4 `template<class T> bool Ipopt::CachedResults< T >::GetCachedResult (T & retResult, const std::vector< const TaggedObject * > & dependents) const`

Method for retrieving a cached result, providing only a std::vector of TaggedObjects.

Definition at line 585 of file IpCachedResults.hpp.

6.12.3.5 `template<class T> void Ipopt::CachedResults< T >::AddCachedResult1Dep (const T & result, const TaggedObject * dependent1)`

Method for adding a result to the cache, proving one dependency as a [TaggedObject](#) explicitly.

Definition at line 593 of file IpCachedResults.hpp.

6.12.3.6 `template<class T> bool Ipopt::CachedResults< T >::GetCachedResult1Dep (T & retResult, const TaggedObject * dependent1)`

Method for retrieving a cached result, proving one dependency as a [TaggedObject](#) explicitly.

Definition at line 607 of file IpCachedResults.hpp.

6.12.3.7 `template<class T> void Ipopt::CachedResults< T >::AddCachedResult2Dep (const T & result, const TaggedObject * dependent1, const TaggedObject * dependent2)`

Method for adding a result to the cache, proving two dependencies as a [TaggedObject](#) explicitly.

Definition at line 620 of file IpCachedResults.hpp.

6.12.3.8 `template<class T> bool Ipopt::CachedResults< T >::GetCachedResult2Dep (T & retResult, const TaggedObject * dependent1, const TaggedObject * dependent2)`

Method for retrieving a cached result, proving two dependencies as a [TaggedObject](#) explicitly.

Definition at line 636 of file IpCachedResults.hpp.

6.12.3.9 `template<class T> void Ipopt::CachedResults< T >::AddCachedResult3Dep (const T & result, const TaggedObject * dependent1, const TaggedObject * dependent2, const TaggedObject * dependent3)`

Method for adding a result to the cache, proving three dependencies as a [TaggedObject](#) explicitly.

Definition at line 650 of file IpCachedResults.hpp.

6.12.3.10 `template<class T> bool Ipopt::CachedResults< T >::GetCachedResult3Dep (T & retResult, const TaggedObject * dependent1, const TaggedObject * dependent2, const TaggedObject * dependent3)`

Method for retrieving a cached result, proving three dependencies as a [TaggedObject](#) explicitly.

Definition at line 668 of file IpCachedResults.hpp.

6.12.3.11 `template<class T> bool Ipopt::CachedResults< T >::GetCachedResult1Dep (T & retResult, const TaggedObject & dependent1) [inline]`

Definition at line 167 of file IpCachedResults.hpp.

6.12.3.12 `template<class T> bool Ipopt::CachedResults< T >::GetCachedResult2Dep (T & retResult, const TaggedObject & dependent1, const TaggedObject & dependent2) [inline]`

Definition at line 171 of file IpCachedResults.hpp.

6.12.3.13 `template<class T> bool Ipopt::CachedResults< T >::GetCachedResult3Dep (T & retResult, const TaggedObject & dependent1, const TaggedObject & dependent2, const TaggedObject & dependent3) [inline]`

Definition at line 177 of file IpCachedResults.hpp.

6.12.3.14 `template<class T> void Ipopt::CachedResults< T >::AddCachedResult1Dep (const T & result, const TaggedObject & dependent1) [inline]`

Definition at line 184 of file IpCachedResults.hpp.

6.12.3.15 `template<class T> void Ipopt::CachedResults< T >::AddCachedResult2Dep (const T & result, const TaggedObject & dependent1, const TaggedObject & dependent2) [inline]`

Definition at line 189 of file IpCachedResults.hpp.

6.12.3.16 `template<class T> void Ipopt::CachedResults< T >::AddCachedResult3Dep (const T & result, const TaggedObject & dependent1, const TaggedObject & dependent2, const TaggedObject & dependent3) [inline]`

Definition at line 195 of file IpCachedResults.hpp.

6.12.3.17 `template<class T> bool Ipopt::CachedResults< T >::InvalidateResult (const std::vector< const TaggedObject * > & dependents, const std::vector< Number > & scalar_dependents)`

Invalidates the result for given dependencies.

Sets the stale flag for the corresponding cached result to true if it is found. Returns true, if the result was found.

Definition at line 685 of file IpCachedResults.hpp.

6.12.3.18 `template<class T> void Ipopt::CachedResults< T >::Clear ()`

Invalidates all cached results.

Definition at line 708 of file IpCachedResults.hpp.

6.12.3.19 `template<class T> void Ipopt::CachedResults< T >::Clear (Int max_cache_size)`

Invalidate all cached results and changes max_cache_size.

Definition at line 723 of file IpCachedResults.hpp.

6.12.3.20 `template<class T> void Ipopt::CachedResults< T >::operator= (const CachedResults< T > &) [private]`

Overloaded Equals Operator.

6.12.3.21 `template<class T> void Ipopt::CachedResults< T >::CleanupInvalidatedResults () const [private]`

internal method for removing stale DependentResults from the list.

It is called at the beginning of every GetDependentResult method.

Definition at line 730 of file IpCachedResults.hpp.

6.12.3.22 `template<class T> void Ipopt::CachedResults< T >::DebugPrintCachedResults () const` [private]

Print list of currently cached results.

Definition at line 757 of file IpCachedResults.hpp.

6.12.4 Member Data Documentation

6.12.4.1 `template<class T> Int Ipopt::CachedResults< T >::max_cache_size_` [private]

maximum number of cached results

Definition at line 236 of file IpCachedResults.hpp.

6.12.4.2 `template<class T> std::list<DependentResult<T>*>* Ipopt::CachedResults< T >::cached_results_` [mutable], [private]

list of currently cached results.

Definition at line 239 of file IpCachedResults.hpp.

The documentation for this class was generated from the following file:

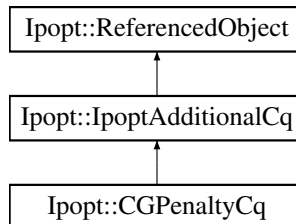
- Common/[IpCachedResults.hpp](#)

6.13 Ipopt::CGPenaltyCq Class Reference

Class for all Chen-Goldfarb penalty method specific calculated quantities.

```
#include <IpCGPenaltyCq.hpp>
```

Inheritance diagram for Ipopt::CGPenaltyCq:



Public Member Functions

- bool [Initialize](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)
This method must be called to initialize the global algorithmic parameters.

Constructors/Destructors

- [CGPenaltyCq](#) ([IpoptNLP](#) *ip_nlp, [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cg)
Constructor.
- virtual [~CGPenaltyCq](#) ()
Default destructor.

Methods for the Chen-Goldfarb line search

- [Number curr_jac_cd_norm](#) (Index nrm_type)
Compute $\| \text{delta}_c, \text{delta}_d \|_{\infty}$.
- [Number curr_scaled_y_Amax](#) ()
Compute gradient scaling based $y \rightarrow A_{\max}$.
- [Number curr_added_y_nrm2](#) ()
Compute the 2-norm of y plus delta_y .
- [Number curr_penalty_function](#) ()
Method for the penalty function at current point.
- [Number trial_penalty_function](#) ()
Method for the penalty function at trial point.
- [Number curr_direct_deriv_penalty_function](#) ()
Method for the directional derivative of the penalty function at current point with current step in delta .
- [Number curr_fast_direct_deriv_penalty_function](#) ()
Method for the directional derivative of the penalty function at current point with current "fast" step in $\text{delta}_{\text{cgpen}}$.
- [Number dT_times_barH_times_d](#) ()
Quality of $d^T \text{Aug}(H) d$.
- [Number curr_cg_pert_fact](#) ()
Method for the current value for the perturbation factor for the Chen-Goldfarb method.
- [Number compute_curr_cg_penalty](#) (const Number)
Method for choose line search penalty parameter.
- [Number compute_curr_cg_penalty_scale](#) ()
Method for choose penalty parameters for scaling the KKT system.

Static Public Member Functions

- static void [RegisterOptions](#) (const SmartPtr< [RegisteredOptions](#) > &roptions)
Methods for IpoptType.

Private Member Functions

- [CGPenaltyData](#) & [CGPenData](#) ()
Method to easily access CGPenalty data.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CGPenaltyCq](#) ()
Default Constructor.
- [CGPenaltyCq](#) (const [CGPenaltyCq](#) &)
Copy Constructor.
- void [operator=](#) (const [CGPenaltyCq](#) &)
Overloaded Equals Operator.

Private Attributes

- bool [initialize_called_](#)
flag indicating if Initialize method has been called (for debugging)

Pointers for easy access to data and NLP information. To

avoid circular references of Smart Pointers, we use a regular pointer here.

- [IpoptNLP * ip_nlp_](#)
- [IpoptData * ip_data_](#)
- [IpoptCalculatedQuantities * ip_cq_](#)

Caches for the Chen-Goldfarb line search

- [CachedResults< Number > curr_fast_direct_deriv_penalty_function_cache_](#)
- [CachedResults< Number > curr_jac_cd_norm_cache_](#)
- [CachedResults< Number > curr_scaled_y_Amax_cache_](#)
- [CachedResults< Number > curr_added_y_nrm2_cache_](#)
- [CachedResults< Number > curr_penalty_function_cache_](#)
Cache for the penalty function at current point.
- [CachedResults< Number > trial_penalty_function_cache_](#)
Cache for the penalty function at trial point.
- [CachedResults< Number > curr_direct_deriv_penalty_function_cache_](#)
Cache for the directional derivative of the penalty function at current point with step in delta.
- [CachedResults< Number > curr_cg_pert_fact_cache_](#)
Cache for Chen-Goldfarb perturbation factor.
- [Number reference_infeasibility_](#)
Parameters for penalty method.

6.13.1 Detailed Description

Class for all Chen-Goldfarb penalty method specific calculated quantities.

Definition at line 22 of file IpCGPenaltyCq.hpp.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 `Ipopt::CGPenaltyCq::CGPenaltyCq (IpoptNLP * ip_nlp, IpoptData * ip_data, IpoptCalculatedQuantities * ip_cg)`

Constructor.

6.13.2.2 `virtual Ipopt::CGPenaltyCq::~~CGPenaltyCq () [virtual]`

Default destructor.

6.13.2.3 `Ipopt::CGPenaltyCq::CGPenaltyCq () [private]`

Default Constructor.

6.13.2.4 `Ipopt::CGPenaltyCq::CGPenaltyCq (const CGPenaltyCq &) [private]`

Copy Constructor.

6.13.3 Member Function Documentation

6.13.3.1 `bool Ipopt::CGPenaltyCq::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [virtual]`

This method must be called to initialize the global algorithmic parameters.

The parameters are taken from the [OptionsList](#) object.

Implements [Ipopt::IpoptAdditionalCq](#).

6.13.3.2 Number Ipopt::CGPenaltyCq::curr_jac_cd_norm (Index *nrm_type*)

Compute $\|\delta_c, \delta_d\|_{\infty}$.

6.13.3.3 Number Ipopt::CGPenaltyCq::curr_scaled_y_Amax ()

Compute gradient scaling based $y \rightarrow A_{\max}$.

6.13.3.4 Number Ipopt::CGPenaltyCq::curr_added_y_nrm2 ()

Compute the 2-norm of y plus δy .

6.13.3.5 Number Ipopt::CGPenaltyCq::curr_penalty_function ()

Method for the penalty function at current point.

6.13.3.6 Number Ipopt::CGPenaltyCq::trial_penalty_function ()

Method for the penalty function at trial point.

6.13.3.7 Number Ipopt::CGPenaltyCq::curr_direct_deriv_penalty_function ()

Method for the directional derivative of the penalty function at current point with current step in δ .

6.13.3.8 Number Ipopt::CGPenaltyCq::curr_fast_direct_deriv_penalty_function ()

Method for the directional derivative of the penalty function at current point with current "fast" step in δ_{cgpen} .

6.13.3.9 Number Ipopt::CGPenaltyCq::dT_times_barH_times_d ()

Quality of $d^T A_{\text{ug}}(H) d$.

6.13.3.10 Number Ipopt::CGPenaltyCq::curr_cg_pert_fact ()

Method for the current value for the perturbation factor for the Chen-Goldfarb method.

The factor is computed as 2-norm of the constraints divided by the current penalty parameter

6.13.3.11 Number Ipopt::CGPenaltyCq::compute_curr_cg_penalty (const Number)

Method for choose line search penalty parameter.

6.13.3.12 Number Ipopt::CGPenaltyCq::compute_curr_cg_penalty_scale ()

Method for choose penalty parameters for scaling the KKT system.

6.13.3.13 static void Ipopt::CGPenaltyCq::RegisterOptions (const SmartPtr< RegisteredOptions > & roptions)
[static]

Methods for IpoptType.

6.13.3.14 void Ipopt::CGPenaltyCq::operator= (const CGPenaltyCq &) [private]

Overloaded Equals Operator.

6.13.3.15 CGPenaltyData& Ipopt::CGPenaltyCq::CGPenData () [inline], [private]

Method to easily access CGPenalty data.

Definition at line 115 of file IpCGPenaltyCq.hpp.

6.13.4 Member Data Documentation**6.13.4.1 IpoptNLP* Ipopt::CGPenaltyCq::ip_nlp_ [private]**

Definition at line 109 of file IpCGPenaltyCq.hpp.

6.13.4.2 IpoptData* Ipopt::CGPenaltyCq::ip_data_ [private]

Definition at line 110 of file IpCGPenaltyCq.hpp.

6.13.4.3 IpoptCalculatedQuantities* Ipopt::CGPenaltyCq::ip_cq_ [private]

Definition at line 111 of file IpCGPenaltyCq.hpp.

6.13.4.4 CachedResults<Number> Ipopt::CGPenaltyCq::curr_fast_direct_deriv_penalty_function_cache_ [private]

Definition at line 125 of file IpCGPenaltyCq.hpp.

6.13.4.5 CachedResults<Number> Ipopt::CGPenaltyCq::curr_jac_cd_norm_cache_ [private]

Definition at line 126 of file IpCGPenaltyCq.hpp.

6.13.4.6 CachedResults<Number> Ipopt::CGPenaltyCq::curr_scaled_y_Amax_cache_ [private]

Definition at line 127 of file IpCGPenaltyCq.hpp.

6.13.4.7 CachedResults<Number> Ipopt::CGPenaltyCq::curr_added_y_nrm2_cache_ [private]

Definition at line 128 of file IpCGPenaltyCq.hpp.

6.13.4.8 CachedResults<Number> Ipopt::CGPenaltyCq::curr_penalty_function_cache_ [private]

Cache for the penalty function at current point.

Definition at line 130 of file IpCGPenaltyCq.hpp.

6.13.4.9 CachedResults<Number> Ipopt::CGPenaltyCq::trial_penalty_function_cache_ [private]

Cache for the penalty function at trial point.

Definition at line 132 of file IpCGPenaltyCq.hpp.

6.13.4.10 CachedResults<Number> Ipopt::CGPenaltyCq::curr_direct_deriv_penalty_function_cache_ [private]

Cache for the directional derivative of the penalty function at current point with step in delta.

Definition at line 135 of file IpCGPenaltyCq.hpp.

6.13.4.11 CachedResults<Number> Ipopt::CGPenaltyCq::curr_cg_pert_fact_cache_ [private]

Cache for Chen-Goldfarb perturbation factor.

Definition at line 137 of file IpCGPenaltyCq.hpp.

6.13.4.12 `Number Ipopt::CGPenaltyCq::reference_infeasibility_ [private]`

Parameters for penalty method.

Definition at line 142 of file `IpCGPenaltyCq.hpp`.

6.13.4.13 `bool Ipopt::CGPenaltyCq::initialize_called_ [private]`

flag indicating if Initialize method has been called (for debugging)

Definition at line 147 of file `IpCGPenaltyCq.hpp`.

The documentation for this class was generated from the following file:

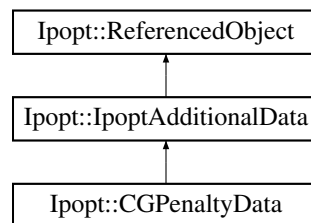
- [contrib/CGPenalty/IpCGPenaltyCq.hpp](#)

6.14 `Ipopt::CGPenaltyData` Class Reference

Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm.

```
#include <IpCGPenaltyData.hpp>
```

Inheritance diagram for `Ipopt::CGPenaltyData`:



Public Member Functions

- `bool Initialize (const Journalist &jnlst, const OptionsList &options, const std::string &prefix)`
This method must be called to initialize the global algorithmic parameters.
- `bool InitializeDataStructures ()`
Initialize Data Structures.
- `SmartPtr< const IteratesVector > delta_cgpen () const`
Delta for the Chen-Goldfarb search direction.
- `void set_delta_cgpen (SmartPtr< IteratesVector > &delta_pen)`
Set the delta_cgpen - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.
- `void set_delta_cgpen (SmartPtr< const IteratesVector > &delta_pen)`
Set the delta_cgpen - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.
- `SmartPtr< const IteratesVector > delta_cgfast () const`
Delta for the fast Chen-Goldfarb search direction.
- `void set_delta_cgfast (SmartPtr< IteratesVector > &delta_fast)`
Set the delta_cgpen - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.
- `Number CurrPenaltyPert ()`

- void [SetCurrPenaltyPert](#) ([Number](#) curr_penalty_pert)
- void [SetNeverTryPureNewton](#) (bool never_try_pure_Newton)
- [Index](#) [NeverTryPureNewton](#) ()
- [Index](#) [restor_iter](#) ()
- void [SetRestorIter](#) ([Index](#) restor_iter)
- [Number](#) [restor_counter](#) ()
- void [SetRestorCounter](#) ([Number](#) restor_counter)
- void [SetPrimalStepSize](#) ([Number](#) max_alpha_x)
- [Number](#) [PrimalStepSize](#) ()
- [Number](#) [curr_penalty](#) () const
- void [Set_penalty](#) ([Number](#) penalty)
- void [SetPenaltyUninitialized](#) ()
- bool [PenaltyInitialized](#) () const
- [Number](#) [curr_kkt_penalty](#) () const
- void [Set_kkt_penalty](#) ([Number](#) kkt_penalty)
- void [SetKKTPenaltyUninitialized](#) ()
- bool [KKTPenaltyInitialized](#) () const

Constructors/Destructors

- [CGPenaltyData](#) ()
Constructor.
- [~CGPenaltyData](#) ()
Default destructor.

Chen-Goldfarb step2. Those fields can be used to store
directions related to the Chen-Goldfarb algorithm

- bool [HaveCgPenDeltas](#) () const
- void [SetHaveCgPenDeltas](#) (bool have_cgpen_deltas)
- bool [HaveCgFastDeltas](#) () const
- void [SetHaveCgFastDeltas](#) (bool have_cgfast_deltas)

Public Methods for updating iterates

- void [AcceptTrialPoint](#) ()
Set the current iterate values from the trial values.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CGPenaltyData](#) (const [CGPenaltyData](#) &)
Copy Constructor.
- void [operator=](#) (const [CGPenaltyData](#) &)
Overloaded Equals Operator.

Private Attributes

- bool `initialize_called_`
flag indicating if Initialize method has been called (for debugging)

Pure Chen-Goldfarb step for the penatly function. This

used to transfer the information about the step from the computation of the overall search direction to the line search.

- `SmartPtr< const IteratesVector > delta_cgpen_`
- bool `have_cgpen_deltas_`
The following flag is set to true, if some other part of the algorithm has already computed the Chen-Goldfarb step.

Fast Chen-Goldfarb step for the penatly function. This

used to transfer the information about the step from the computation of the overall search direction to the line search.

- `SmartPtr< const IteratesVector > delta_cgfast_`
- bool `have_cgfast_deltas_`
The following flag is set to true, if some other part of the algorithm has already computed the fast Chen-Goldfarb step.

penalty method

- bool `never_try_pure_Newton_`
Flag indicating whether the pure Newton method is used.
- `Index restor_iter_`
The iteration at which pure Newton method is given up.
- `Number restor_counter_`

penalty parameters

- `Number curr_penalty_`
- bool `penalty_initialized_`
- `Number curr_kkt_penalty_`
- bool `kkt_penalty_initialized_`
- `Number curr_penalty_pert_`
- `Number max_alpha_x_`

6.14.1 Detailed Description

Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm.

Definition at line 22 of file `IpCGPenaltyData.hpp`.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `Ipopt::CGPenaltyData::CGPenaltyData ()`

Constructor.

6.14.2.2 `Ipopt::CGPenaltyData::~~CGPenaltyData ()`

Default destructor.

6.14.2.3 `Ipopt::CGPenaltyData::CGPenaltyData (const CGPenaltyData &) [private]`

Copy Constructor.

6.14.3 Member Function Documentation

6.14.3.1 `bool Ipopt::CGPenaltyData::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [virtual]`

This method must be called to initialize the global algorithmic parameters.

The parameters are taken from the [OptionsList](#) object.

Implements [Ipopt::IpoptAdditionalData](#).

6.14.3.2 `bool Ipopt::CGPenaltyData::InitializeDataStructures () [virtual]`

Initialize Data Structures.

Implements [Ipopt::IpoptAdditionalData](#).

6.14.3.3 `SmartPtr< const IteratesVector > Ipopt::CGPenaltyData::delta_cgpen () const [inline]`

Delta for the Chen-Goldfarb search direction.

Definition at line 264 of file IpCGPenaltyData.hpp.

6.14.3.4 `void Ipopt::CGPenaltyData::set_delta_cgpen (SmartPtr< IteratesVector > & delta_pen) [inline]`

Set the delta_cgpen - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.

Definition at line 280 of file IpCGPenaltyData.hpp.

6.14.3.5 `void Ipopt::CGPenaltyData::set_delta_cgpen (SmartPtr< const IteratesVector > & delta_pen) [inline]`

Set the delta_cgpen - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.

This is the version that is happy with a pointer to const [IteratesVector](#).

Definition at line 299 of file IpCGPenaltyData.hpp.

6.14.3.6 `SmartPtr< const IteratesVector > Ipopt::CGPenaltyData::delta_cgfast () const [inline]`

Delta for the fast Chen-Goldfarb search direction.

Definition at line 272 of file IpCGPenaltyData.hpp.

6.14.3.7 `void Ipopt::CGPenaltyData::set_delta_cgfast (SmartPtr< IteratesVector > & delta_fast) [inline]`

Set the delta_cgpen - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.

Definition at line 318 of file IpCGPenaltyData.hpp.

6.14.3.8 `bool Ipopt::CGPenaltyData::HaveCgPenDeltas () const [inline]`

Definition at line 73 of file IpCGPenaltyData.hpp.

6.14.3.9 `void Ipopt::CGPenaltyData::SetHaveCgPenDeltas (bool have_cgpen_deltas) [inline]`

Definition at line 77 of file IpCGPenaltyData.hpp.

6.14.3.10 `bool Ipopt::CGPenaltyData::HaveCgFastDeltas () const [inline]`

Definition at line 82 of file `IpCGPenaltyData.hpp`.

6.14.3.11 `void Ipopt::CGPenaltyData::SetHaveCgFastDeltas (bool have_cgfast_deltas) [inline]`

Definition at line 86 of file `IpCGPenaltyData.hpp`.

6.14.3.12 `void Ipopt::CGPenaltyData::AcceptTrialPoint () [virtual]`

Set the current iterate values from the trial values.

Implements [Ipopt::IpoptAdditionalData](#).

6.14.3.13 `Number Ipopt::CGPenaltyData::CurrPenaltyPert () [inline]`

Definition at line 99 of file `IpCGPenaltyData.hpp`.

6.14.3.14 `void Ipopt::CGPenaltyData::SetCurrPenaltyPert (Number curr_penalty_pert) [inline]`

Definition at line 103 of file `IpCGPenaltyData.hpp`.

6.14.3.15 `void Ipopt::CGPenaltyData::SetNeverTryPureNewton (bool never_try_pure_Newton) [inline]`

Definition at line 108 of file `IpCGPenaltyData.hpp`.

6.14.3.16 `Index Ipopt::CGPenaltyData::NeverTryPureNewton () [inline]`

Definition at line 112 of file `IpCGPenaltyData.hpp`.

6.14.3.17 `Index Ipopt::CGPenaltyData::restor_iter () [inline]`

Definition at line 117 of file `IpCGPenaltyData.hpp`.

6.14.3.18 `void Ipopt::CGPenaltyData::SetRestorIter (Index restor_iter) [inline]`

Definition at line 122 of file `IpCGPenaltyData.hpp`.

6.14.3.19 `Number Ipopt::CGPenaltyData::restor_counter () [inline]`

Definition at line 126 of file `IpCGPenaltyData.hpp`.

6.14.3.20 `void Ipopt::CGPenaltyData::SetRestorCounter (Number restor_counter) [inline]`

Definition at line 130 of file `IpCGPenaltyData.hpp`.

6.14.3.21 `void Ipopt::CGPenaltyData::SetPrimalStepSize (Number max_alpha_x) [inline]`

Definition at line 135 of file `IpCGPenaltyData.hpp`.

6.14.3.22 `Number Ipopt::CGPenaltyData::PrimalStepSize () [inline]`

Definition at line 139 of file `IpCGPenaltyData.hpp`.

6.14.3.23 `Number Ipopt::CGPenaltyData::curr_penalty () const [inline]`

Definition at line 144 of file `IpCGPenaltyData.hpp`.

6.14.3.24 `void Ipopt::CGPenaltyData::Set_penalty (Number penalty) [inline]`

Definition at line 149 of file IpCGPenaltyData.hpp.

6.14.3.25 `void Ipopt::CGPenaltyData::SetPenaltyUninitialized () [inline]`

Definition at line 154 of file IpCGPenaltyData.hpp.

6.14.3.26 `bool Ipopt::CGPenaltyData::PenaltyInitialized () const [inline]`

Definition at line 158 of file IpCGPenaltyData.hpp.

6.14.3.27 `Number Ipopt::CGPenaltyData::curr_kkt_penalty () const [inline]`

Definition at line 162 of file IpCGPenaltyData.hpp.

6.14.3.28 `void Ipopt::CGPenaltyData::Set_kkt_penalty (Number kkt_penalty) [inline]`

Definition at line 167 of file IpCGPenaltyData.hpp.

6.14.3.29 `void Ipopt::CGPenaltyData::SetKKTPenaltyUninitialized () [inline]`

Definition at line 172 of file IpCGPenaltyData.hpp.

6.14.3.30 `bool Ipopt::CGPenaltyData::KKTPenaltyInitialized () const [inline]`

Definition at line 176 of file IpCGPenaltyData.hpp.

6.14.3.31 `void Ipopt::CGPenaltyData::operator= (const CGPenaltyData &) [private]`

Overloaded Equals Operator.

6.14.4 Member Data Documentation

6.14.4.1 `SmartPtr<const IteratesVector> Ipopt::CGPenaltyData::delta_cgpen_ [private]`

Definition at line 189 of file IpCGPenaltyData.hpp.

6.14.4.2 `bool Ipopt::CGPenaltyData::have_cgpen_deltas_ [private]`

The following flag is set to true, if some other part of the algorithm has already computed the Chen-Goldfarb step.

This flag is reset when the AcceptTrialPoint method is called. ToDo: we could cue off of a null delta_cgpen_;

Definition at line 195 of file IpCGPenaltyData.hpp.

6.14.4.3 `SmartPtr<const IteratesVector> Ipopt::CGPenaltyData::delta_cgfast_ [private]`

Definition at line 203 of file IpCGPenaltyData.hpp.

6.14.4.4 `bool Ipopt::CGPenaltyData::have_cgfast_deltas_ [private]`

The following flag is set to true, if some other part of the algorithm has already computed the fast Chen-Goldfarb step.

This flag is reset when the AcceptTrialPoint method is called.

- ToDo: we could cue off of a null delta_cgfast_;

Definition at line 209 of file IpCGPenaltyData.hpp.

6.14.4.5 `bool Ipopt::CGPenaltyData::never_try_pure_Newton_ [private]`

Flag indicating whether the pure Newton method is used.

Definition at line 215 of file IpCGPenaltyData.hpp.

6.14.4.6 `Index Ipopt::CGPenaltyData::restor_iter_ [private]`

The iteration at which pure Newton method is given up.

Definition at line 218 of file IpCGPenaltyData.hpp.

6.14.4.7 `Number Ipopt::CGPenaltyData::restor_counter_ [private]`

Definition at line 219 of file IpCGPenaltyData.hpp.

6.14.4.8 `Number Ipopt::CGPenaltyData::curr_penalty_ [private]`

Definition at line 222 of file IpCGPenaltyData.hpp.

6.14.4.9 `bool Ipopt::CGPenaltyData::penalty_initialized_ [private]`

Definition at line 223 of file IpCGPenaltyData.hpp.

6.14.4.10 `Number Ipopt::CGPenaltyData::curr_kkt_penalty_ [private]`

Definition at line 224 of file IpCGPenaltyData.hpp.

6.14.4.11 `bool Ipopt::CGPenaltyData::kkt_penalty_initialized_ [private]`

Definition at line 225 of file IpCGPenaltyData.hpp.

6.14.4.12 `Number Ipopt::CGPenaltyData::curr_penalty_pert_ [private]`

Definition at line 226 of file IpCGPenaltyData.hpp.

6.14.4.13 `Number Ipopt::CGPenaltyData::max_alpha_x_ [private]`

Definition at line 227 of file IpCGPenaltyData.hpp.

6.14.4.14 `bool Ipopt::CGPenaltyData::initialize_called_ [private]`

flag indicating if Initialize method has been called (for debugging)

Definition at line 232 of file IpCGPenaltyData.hpp.

The documentation for this class was generated from the following file:

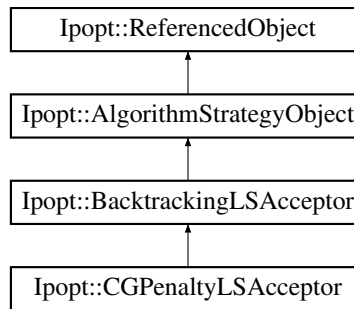
- contrib/CGPenalty/[IpCGPenaltyData.hpp](#)

6.15 Ipopt::CGPenaltyLSAcceptor Class Reference

Line search acceptor, based on the Chen-Goldfarb penalty function approach.

```
#include <IpCGPenaltyLSAcceptor.hpp>
```

Inheritance diagram for Ipopt::CGPenaltyLSAcceptor:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
InitializeImpl - overloaded from [AlgorithmStrategyObject](#).
- virtual void [Reset](#) ()
Reset the acceptor.
- virtual void [InitThisLineSearch](#) (bool in_watchdog)
Initialization for the next line search.
- virtual void [PrepareRestoPhaseStart](#) ()
Method that is called before the restoration phase is called.
- virtual [Number](#) [CalculateAlphaMin](#) ()
Method returning the lower bound on the trial step sizes.
- virtual bool [CheckAcceptabilityOfTrialPoint](#) ([Number](#) alpha_primal)
Method for checking if current trial point is acceptable.
- virtual bool [TrySecondOrderCorrection](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try a second order correction for the constraints.
- virtual bool [TryCorrector](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try higher order corrector (for fast local convergence).
- virtual char [UpdateForNextIteration](#) ([Number](#) alpha_primal_test)
Method for ending the current line search.
- virtual void [StartWatchDog](#) ()
Method for setting internal data if the watchdog procedure is started.
- virtual void [StopWatchDog](#) ()
Method for setting internal data if the watchdog procedure is stopped.
- virtual bool [RestoredIterate](#) ()
Method for telling the [BacktrackingLineSearch](#) object that a previous iterate has been restored.
- virtual bool [NeverRestorationPhase](#) ()
Method for telling the [BacktrackingLineSearch](#) object that the restoration is not needed.
- virtual bool [DoFallback](#) ()
Method for doing a fallback approach in case no search direction could be computed.

Constructors/Destructors

- [CGPenaltyLSAcceptor](#) (const [SmartPtr](#)< [PDSolver](#) > &pd_solver)
Constructor.
- virtual [~CGPenaltyLSAcceptor](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for [OptionsList](#).

Private Member Functions

- [CGPenaltyData](#) & [CGPenData](#) ()
Method to easily access CGPenalty data.
- [CGPenaltyCq](#) & [CGPenCq](#) ()
Method to easily access CGPenalty calculated quantities.
- bool [IsAcceptableToPiecewisePenalty](#) ([Number](#) alpha_primal_test)
Check if the trial point is acceptable to the piecewise penalty list.
- bool [ArmijoHolds](#) ([Number](#) alpha_primal_test)
Check if the trial point is acceptable by the Armijo condition.
- bool [CurrentIsBest](#) ()
- void [StoreBestPoint](#) ()
- bool [RestoreBestPoint](#) ()
- bool [MultipliersDiverged](#) ()
- char [UpdatePenaltyParameter](#) ()

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CGPenaltyLSAcceptor](#) (const [CGPenaltyLSAcceptor](#) &)
Copy Constructor.
- void [operator=](#) (const [CGPenaltyLSAcceptor](#) &)
Overloaded Equals Operator.

Static Private Member Functions

- static bool [Compare_le](#) ([Number](#) lhs, [Number](#) rhs, [Number](#) BasVal)
Check comparison " $lhs \leq rhs$ ", using machine precision based on BasVal.

Private Attributes

- [Number](#) pen_curr_mu_
- [Number](#) theta_min_
Parameters deciding when the piecewise penalty acceptor shall be closed.
- bool [accepted_by_Armijo](#)_
Flag indicating whether the trial point is accepted by the Armijo condition or the PLPF condition.
- [Number](#) min_alpha_primal_
Min step size that triggers nonmonotone method.
- [Index](#) max_soc_
Maximal number of second order correction steps.
- [Number](#) kappa_soc_

Required reduction in constraint violation before trying multiple second order correction steps κ_{soc} .

- [Index counter_first_type_penalty_updates_](#)
Counter for increases of penalty parameter.
- [Index counter_second_type_penalty_updates_](#)
- [Number curr_eta_](#)
eta parameter
- [Index ls_counter_](#)
counter for cut backs in the line search
- [Number best_KKT_error_](#)
Record the lease KKT error found so far.
- [SmartPtr< const IteratesVector > best_iterate_](#)
Store the iterate with best KKT error found so far.
- [Number mult_diverg_feasibility_tol_](#)
Check if the multipliers are diverging.
- [Number mult_diverg_y_tol_](#)
- [bool never_use_piecewise_penalty_ls_](#)
Flag for whether or not use piecewise penalty line search.
- [PiecewisePenalty PiecewisePenalty_](#)
- [bool reset_piecewise_penalty_](#)
Flag indicating whether [PiecewisePenalty](#) has to be initialized.
- [Index jump_for_tiny_step_](#)

Parameters for the penalty function algorithm.

- [Number eta_penalty_](#)
Relaxation factor in the Armijo condition for the penalty function.
- [Number penalty_update_infeasibility_tol_](#)
Tolerance for infeasibility part in penalty parameter update rule.
- [Number eta_min_](#)
Minimal tolerance for step part in penalty parameter update rule.
- [Number penalty_update_compl_tol_](#)
Tolerance for complementarity part in penalty parameter update rule.
- [Number chi_hat_](#)
- [Number chi_tilde_](#)
- [Number chi_cup_](#)
- [Number gamma_hat_](#)
- [Number gamma_tilde_](#)
- [Number penalty_max_](#)
- [Number epsilon_c_](#)
- [Number piecewisepenalty_gamma_obj_](#)
Parameters for piecewise penalty acceptor.
- [Number piecewisepenalty_gamma_infeasi_](#)
- [Number pen_theta_max_](#)
Upper bound on infeasibility.
- [Number pen_theta_max_fact_](#)
- [Number reference_theta_](#)

Information related to watchdog procedure

- [Number reference_penalty_function_](#)
Penalty function at the point with respect to which progress is to be made.

- [Number reference_direct_deriv_penalty_function_](#)
Directional derivative of penalty function at the point with respect to which progress is to be made.
- [Number reference_curr_direct_f_nrm_](#)
- [Number watchdog_penalty_function_](#)
Penalty function at the point with respect to which progress is to be made (at watchdog point)
- [Number watchdog_direct_deriv_penalty_function_](#)
Directional derivative of penalty function at the point with respect to which progress is to be made (at watchdog point)
- [SmartPtr< const IteratesVector > watchdog_delta_cgpen_](#)
Backup for the Chen-Goldfarb search direction (needed in the update rule for the penalty parameter.

Strategy objective that are used

- [SmartPtr< PDSystemSolver > pd_solver_](#)

Additional Inherited Members

6.15.1 Detailed Description

Line search acceptor, based on the Chen-Goldfarb penalty function approach.

Definition at line 23 of file IpCGPenaltyLSAcceptor.hpp.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 `Ipopt::CGPenaltyLSAcceptor::CGPenaltyLSAcceptor (const SmartPtr< PDSystemSolver > & pd_solver)`

Constructor.

The [PDSystemSolver](#) object only needs to be provided (i.e. not NULL) if second order correction or corrector steps are to be used.

6.15.2.2 `virtual Ipopt::CGPenaltyLSAcceptor::~~CGPenaltyLSAcceptor () [virtual]`

Default destructor.

6.15.2.3 `Ipopt::CGPenaltyLSAcceptor::CGPenaltyLSAcceptor (const CGPenaltyLSAcceptor &) [private]`

Copy Constructor.

6.15.3 Member Function Documentation

6.15.3.1 `virtual bool Ipopt::CGPenaltyLSAcceptor::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

InitializeImpl - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.2 `virtual void Ipopt::CGPenaltyLSAcceptor::Reset () [virtual]`

Reset the acceptor.

This function should be called if all previous information should be discarded when the line search is performed the next time. For example, this method should be called if the barrier parameter is changed.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.3 `virtual void Ipopt::CGPenaltyLSAcceptor::InitThisLineSearch (bool in_watchdog) [virtual]`

Initialization for the next line search.

The flag `in_watchdog` indicates if we are currently in an active watchdog procedure.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.4 `virtual void Ipopt::CGPenaltyLSAcceptor::PrepareRestoPhaseStart () [virtual]`

Method that is called before the restoration phase is called.

Here, we can set up things that are required in the termination test for the restoration phase.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.5 `virtual Number Ipopt::CGPenaltyLSAcceptor::CalculateAlphaMin () [virtual]`

Method returning the lower bound on the trial step sizes.

If the backtracking procedure encounters a trial step size below this value after the first trial set, it switches to the (soft) restoration phase.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.6 `virtual bool Ipopt::CGPenaltyLSAcceptor::CheckAcceptabilityOfTrialPoint (Number alpha_primal) [virtual]`

Method for checking if current trial point is acceptable.

It is assumed that the delta information in `ip_data` is the search direction used in criteria. The primal trial point has to be set before the call.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.7 `virtual bool Ipopt::CGPenaltyLSAcceptor::TrySecondOrderCorrection (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta) [virtual]`

Try a second order correction for the constraints.

If the first trial step (with incoming `alpha_primal`) has been reject, this tries up to `max_soc` second order corrections for the constraints. Here, `alpha_primal_test` is the step size that has to be used in the merit function acceptance tests. On output `actual_delta` has been set to the step including the second order correction if it has been accepted, otherwise it is unchanged. If the SOC step has been accepted, `alpha_primal` has the fraction-to-the-boundary value for the SOC step on output. The return value is true, if a SOC step has been accepted.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.8 `virtual bool Ipopt::CGPenaltyLSAcceptor::TryCorrector (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta) [virtual]`

Try higher order corrector (for fast local convergence).

In contrast to a second order correction step, which tries to make an unacceptable point acceptable by improving constraint violation, this corrector step is tried even if the regular primal-dual step is acceptable.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.9 `virtual char Ipopt::CGPenaltyLSAcceptor::UpdateForNextIteration (Number alpha_primal_test) [virtual]`

Method for ending the current line search.

When it is called, the internal data should be updates, e.g., the penalty parameter might be updated. `alpha_primal_test` is the value of alpha that has been used for in the acceptance test earlier.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.10 `virtual void Ipopt::CGPenaltyLSAcceptor::StartWatchDog () [virtual]`

Method for setting internal data if the watchdog procedure is started.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.11 `virtual void Ipopt::CGPenaltyLSAcceptor::StopWatchDog () [virtual]`

Method for setting internal data if the watchdog procedure is stopped.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.12 `virtual bool Ipopt::CGPenaltyLSAcceptor::RestoredIterate () [virtual]`

Method for telling the [BacktrackingLineSearch](#) object that a previous iterate has been restored.

Reimplemented from [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.13 `virtual bool Ipopt::CGPenaltyLSAcceptor::NeverRestorationPhase () [virtual]`

Method for telling the [BacktrackingLineSearch](#) object that the restoration is not needed.

Reimplemented from [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.14 `virtual bool Ipopt::CGPenaltyLSAcceptor::DoFallback () [virtual]`

Method for doing a fallback approach in case no search direction could be computed.

If no such fall back option is available, return false.

Reimplemented from [Ipopt::BacktrackingLSAcceptor](#).

6.15.3.15 `static void Ipopt::CGPenaltyLSAcceptor::RegisterOptions (SmartPtr< RegisteredOptions > options) [static]`

Methods for [OptionsList](#).

6.15.3.16 `void Ipopt::CGPenaltyLSAcceptor::operator= (const CGPenaltyLSAcceptor &) [private]`

Overloaded Equals Operator.

6.15.3.17 `CGPenaltyData& Ipopt::CGPenaltyLSAcceptor::CGPenData () [inline],[private]`

Method to easily access CGPenalty data.

Definition at line 146 of file IpCGPenaltyLSAcceptor.hpp.

6.15.3.18 `CGPenaltyCq& Ipopt::CGPenaltyLSAcceptor::CGPenCq () [inline],[private]`

Method to easily access CGPenalty calculated quantities.

Definition at line 155 of file IpCGPenaltyLSAcceptor.hpp.

6.15.3.19 `bool Ipopt::CGPenaltyLSAcceptor::IsAcceptableToPiecewisePenalty (Number alpha_primal_test) [private]`

Check if the trial point is acceptable to the piecewise penalty list.

6.15.3.20 `bool Ipopt::CGPenaltyLSAcceptor::ArmijoHolds (Number alpha_primal_test) [private]`

Check if the trial point is acceptable by the Armijo condition.

6.15.3.21 `static bool Ipopt::CGPenaltyLSAcceptor::Compare_le (Number lhs, Number rhs, Number BasVal) [static], [private]`

Check comparison " $lhs \leq rhs$ ", using machine precision based on BasVal.

6.15.3.22 `bool Ipopt::CGPenaltyLSAcceptor::CurrentIsBest () [private]`

6.15.3.23 `void Ipopt::CGPenaltyLSAcceptor::StoreBestPoint () [private]`

6.15.3.24 `bool Ipopt::CGPenaltyLSAcceptor::RestoreBestPoint () [private]`

6.15.3.25 `bool Ipopt::CGPenaltyLSAcceptor::MultipliersDiverged () [private]`

6.15.3.26 `char Ipopt::CGPenaltyLSAcceptor::UpdatePenaltyParameter () [private]`

6.15.4 Member Data Documentation

6.15.4.1 `Number Ipopt::CGPenaltyLSAcceptor::eta_penalty_ [private]`

Relaxation factor in the Armijo condition for the penalty function.

Definition at line 183 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.2 `Number Ipopt::CGPenaltyLSAcceptor::penalty_update_infeasibility_tol_ [private]`

Tolerance for infeasibility part in penalty parameter update rule.

Definition at line 186 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.3 `Number Ipopt::CGPenaltyLSAcceptor::eta_min_ [private]`

Minimal tolerance for step part in penalty parameter update rule.

Definition at line 189 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.4 `Number Ipopt::CGPenaltyLSAcceptor::penalty_update_compl_tol_ [private]`

Tolerance for complementarity part in penalty parameter update rule.

Definition at line 192 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.5 `Number Ipopt::CGPenaltyLSAcceptor::chi_hat_ [private]`

Definition at line 193 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.6 `Number Ipopt::CGPenaltyLSAcceptor::chi_tilde_ [private]`

Definition at line 194 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.7 `Number Ipopt::CGPenaltyLSAcceptor::chi_cup_ [private]`

Definition at line 195 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.8 Number Ipopt::CGPenaltyLSAcceptor::gamma_hat_ [private]

Definition at line 196 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.9 Number Ipopt::CGPenaltyLSAcceptor::gamma_tilde_ [private]

Definition at line 197 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.10 Number Ipopt::CGPenaltyLSAcceptor::penalty_max_ [private]

Definition at line 198 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.11 Number Ipopt::CGPenaltyLSAcceptor::epsilon_c_ [private]

Definition at line 199 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.12 Number Ipopt::CGPenaltyLSAcceptor::piecewisepenalty_gamma_obj_ [private]

Parameters for piecewise penalty acceptor.

Definition at line 201 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.13 Number Ipopt::CGPenaltyLSAcceptor::piecewisepenalty_gamma_infeasi_ [private]

Definition at line 202 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.14 Number Ipopt::CGPenaltyLSAcceptor::pen_theta_max_ [private]

Upper bound on infeasibility.

Definition at line 205 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.15 Number Ipopt::CGPenaltyLSAcceptor::pen_theta_max_fact_ [private]

Definition at line 206 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.16 Number Ipopt::CGPenaltyLSAcceptor::pen_curr_mu_ [private]

Definition at line 209 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.17 Number Ipopt::CGPenaltyLSAcceptor::theta_min_ [private]

Parameters deciding when the piecewise penalty acceptor shall be closed.

Definition at line 212 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.18 bool Ipopt::CGPenaltyLSAcceptor::accepted_by_Armijo_ [private]

Flag indicating whether the trial point is accepted by the Armijo condition or the PLPF condition.

Definition at line 217 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.19 Number Ipopt::CGPenaltyLSAcceptor::min_alpha_primal_ [private]

Min step size that triggers nonmonotone method.

Definition at line 220 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.20 Number Ipopt::CGPenaltyLSAcceptor::reference_theta_ [private]

Definition at line 224 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.21 Index Ipopt::CGPenaltyLSAcceptor::max_soc_ [private]

Maximal number of second order correction steps.

Definition at line 227 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.22 Number Ipopt::CGPenaltyLSAcceptor::kappa_soc_ [private]

Required reduction in constraint violation before trying multiple second order correction steps κ_{soc} .

Definition at line 231 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.23 Index Ipopt::CGPenaltyLSAcceptor::counter_first_type_penalty_updates_ [private]

Counter for increases of penalty parameter.

Definition at line 234 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.24 Index Ipopt::CGPenaltyLSAcceptor::counter_second_type_penalty_updates_ [private]

Definition at line 235 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.25 Number Ipopt::CGPenaltyLSAcceptor::curr_eta_ [private]

eta parameter

Definition at line 237 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.26 Index Ipopt::CGPenaltyLSAcceptor::ls_counter_ [private]

counter for cut backs in the line search

Definition at line 240 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.27 Number Ipopt::CGPenaltyLSAcceptor::best_KKT_error_ [private]

Record the lease KKT error found so far.

Definition at line 242 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.28 SmartPtr<const IteratesVector> Ipopt::CGPenaltyLSAcceptor::best_iterate_ [private]

Store the iterate with best KKT error found so far.

Definition at line 244 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.29 Number Ipopt::CGPenaltyLSAcceptor::mult_diverg_feasibility_tol_ [private]

Check if the multipliers are diverging.

Definition at line 246 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.30 Number Ipopt::CGPenaltyLSAcceptor::mult_diverg_y_tol_ [private]

Definition at line 247 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.31 **Number** Ipopt::CGPenaltyLSAcceptor::reference_penalty_function_ [private]

Penalty function at the point with respect to which progress is to be made.

Definition at line 253 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.32 **Number** Ipopt::CGPenaltyLSAcceptor::reference_direct_deriv_penalty_function_ [private]

Directional derivative of penalty function at the point with respect to which progress is to be made.

Definition at line 256 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.33 **Number** Ipopt::CGPenaltyLSAcceptor::reference_curr_direct_f_nrm_ [private]

Definition at line 257 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.34 **Number** Ipopt::CGPenaltyLSAcceptor::watchdog_penalty_function_ [private]

Penalty function at the point with respect to which progress is to be made (at watchdog point)

Definition at line 260 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.35 **Number** Ipopt::CGPenaltyLSAcceptor::watchdog_direct_deriv_penalty_function_ [private]

Directional derivative of penalty function at the point with respect to which progress is to be made (at watchdog point)

Definition at line 263 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.36 **SmartPtr**<const IteratesVector> Ipopt::CGPenaltyLSAcceptor::watchdog_delta_cgpen_ [private]

Backup for the Chen-Goldfarb search direction (needed in the update rule for the penalty parameter.

Definition at line 266 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.37 **bool** Ipopt::CGPenaltyLSAcceptor::never_use_piecewise_penalty_is_ [private]

Flag for whether or not use piecewise penalty line search.

Definition at line 269 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.38 **PiecewisePenalty** Ipopt::CGPenaltyLSAcceptor::PiecewisePenalty_ [private]

Definition at line 271 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.39 **bool** Ipopt::CGPenaltyLSAcceptor::reset_piecewise_penalty_ [private]

Flag indicating whether [PiecewisePenalty](#) has to be initialized.

Definition at line 273 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.40 **Index** Ipopt::CGPenaltyLSAcceptor::jump_for_tiny_step_ [private]

Definition at line 275 of file IpCGPenaltyLSAcceptor.hpp.

6.15.4.41 **SmartPtr**<PDSysolver> Ipopt::CGPenaltyLSAcceptor::pd_solver_ [private]

Definition at line 279 of file IpCGPenaltyLSAcceptor.hpp.

The documentation for this class was generated from the following file:

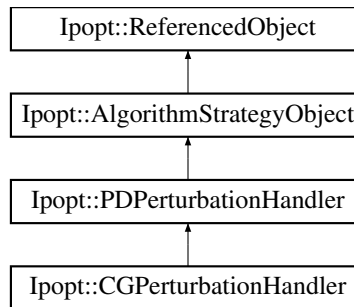
- contrib/CGPenalty/[IpCGPenaltyLSAcceptor.hpp](#)

6.16 Ipopt::CGPerturbationHandler Class Reference

Class for handling the perturbation factors `delta_x`, `delta_s`, `delta_c`, and `delta_d` in the primal dual system.

```
#include <IpCGPerturbationHandler.hpp>
```

Inheritance diagram for Ipopt::CGPerturbationHandler:



Public Member Functions

- virtual bool `InitializeImpl` (const `OptionsList` &options, const std::string &prefix)
Implementation of the initialization method that has to be overloaded by for each derived class.
- bool `ConsiderNewSystem` (`Number` &delta_x, `Number` &delta_s, `Number` &delta_c, `Number` &delta_d)
This method must be called for each new matrix, and before any other method for generating perturbation factors.
- bool `PerturbForSingularity` (`Number` &delta_x, `Number` &delta_s, `Number` &delta_c, `Number` &delta_d)
This method returns perturbation factors for the case when the most recent factorization resulted in a singular matrix.
- bool `PerturbForWrongInertia` (`Number` &delta_x, `Number` &delta_s, `Number` &delta_c, `Number` &delta_d)
This method returns perturbation factors for the case when the most recent factorization resulted in a matrix with an incorrect number of negative eigenvalues.
- void `CurrentPerturbation` (`Number` &delta_x, `Number` &delta_s, `Number` &delta_c, `Number` &delta_d)
Just return the perturbation values that have been determined most recently.

Constructors/Destructors

- `CGPerturbationHandler` ()
Default Constructor.
- virtual `~CGPerturbationHandler` ()
Default destructor.

Static Public Member Functions

- static void `RegisterOptions` (`SmartPtr`< `RegisteredOptions` > roptions)
Methods for IpoptType.

Private Member Functions

- `CGPenaltyData` & `CGPenData` ()
Method to easily access CGPenalty data.
- `CGPenaltyCq` & `CGPenCq` ()

Method to easily access CGPenalty calculated quantities.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `CGPerturbationHandler` (const `CGPerturbationHandler` &)
Copy Constructor.
- void `operator=` (const `CGPerturbationHandler` &)
Overloaded Equals Operator.

Auxiliary methods

- bool `get_deltas_for_wrong_inertia` (`Number` &delta_x, `Number` &delta_s, `Number` &delta_c, `Number` &delta_d)
Internal version of PerturbForWrongInertia with the difference, that finalize_test is not called.
- void `finalize_test` ()
This method is call whenever a matrix had been factorization and is not singular.
- `Number` `delta_cd` ()
Compute perturbation value for constraints.

Private Attributes

- bool `get_deltas_for_wrong_inertia_called_`
Flag indicating if for the given matrix the perturb for wrong inertia method has already been called.
- `Number` `penalty_max_`
The max reference value for scaling the penalty parameter.
- `Number` `mult_diverg_feasibility_tol_`
Feasibility for perturbation in pure Newton method.

Size of the most recent non-zero perturbation.

- `Number` `delta_x_last_`
The last nonzero value for delta_x.
- `Number` `delta_s_last_`
The last nonzero value for delta_s.
- `Number` `delta_c_last_`
The last nonzero value for delta_c.
- `Number` `delta_d_last_`
The last nonzero value for delta_d.

Size of the most recently suggested perturbation for the current matrix.

- `Number` `delta_x_curr_`
The current value for delta_x.
- `Number` `delta_s_curr_`
The current value for delta_s.
- `Number` `delta_c_curr_`
The current value for delta_c.
- `Number` `delta_d_curr_`
The current value for delta_d.

Algorithmic parameters.

- [Number delta_xs_max_](#)
Maximal perturbation for x and s.
- [Number delta_xs_min_](#)
Smallest possible perturbation for x and s.
- [Number delta_xs_first_inc_fact_](#)
Increase factor for delta_xs for first required perturbation.
- [Number delta_xs_inc_fact_](#)
Increase factor for delta_xs for later perturbations.
- [Number delta_xs_dec_fact_](#)
Decrease factor for delta_xs for later perturbations.
- [Number delta_xs_init_](#)
Very first trial value for delta_xs perturbation.
- [Number delta_cd_val_](#)
Size of perturbation for c and d blocks.
- [Number delta_cd_exp_](#)
Exponent on mu in formula for of perturbation for c and d blocks.
- [bool reset_last_](#)
Flag indicating whether the new values are based on the perturbations in the last iteration or in the more recent iteration in which a perturbation was done.
- [Index degen_iters_max_](#)
Required number of iterations for degeneracy conclusions.
- [bool perturb_always_cd_](#)
Flag indicating that the delta_c, delta_d perturbation should always be used.

Handling structural degeneracy

- [enum DegenType { NOT_YET_DETERMINED, NOT_DEGENERATE, DEGENERATE }](#)
Type for degeneracy flags.
- [enum TrialStatus { NO_TEST, TEST_DELTA_C_EQ_0_DELTA_X_EQ_0, TEST_DELTA_C_GT_0_DELTA_X_EQ_0, TEST_DELTA_C_EQ_0_DELTA_X_GT_0, TEST_DELTA_C_GT_0_DELTA_X_GT_0 }](#)
Status of current trial configuration.
- [DegenType hess_degenerate_](#)
Flag indicating whether the reduced Hessian matrix is thought to be structurally singular.
- [DegenType jac_degenerate_](#)
Flag indicating whether the Jacobian of the constraints is thought to be structurally rank-deficient.
- [Index degen_iters_](#)
Flag counting matrices in which degeneracy was observed in the first successive iterations.
- [TrialStatus test_status_](#)
Current status.

Additional Inherited Members**6.16.1 Detailed Description**

Class for handling the perturbation factors delta_x, delta_s, delta_c, and delta_d in the primal dual system.

This class is used by the [PDFullSpaceSolver](#) to handle the cases where the primal-dual system is singular or has the wrong inertia. The perturbation factors are obtained based on simple heuristics, taking into account the size of previous perturbations.

Definition at line 25 of file IpCGPerturbationHandler.hpp.

6.16.2 Member Enumeration Documentation

6.16.2.1 enum `Ipopt::CGPerturbationHandler::DegenType` `[private]`

Type for degeneracy flags.

Enumerator

NOT_YET_DETERMINED

NOT_DEGENERATE

DEGENERATE

Definition at line 140 of file `IpCGPerturbationHandler.hpp`.

6.16.2.2 enum `Ipopt::CGPerturbationHandler::TrialStatus` `[private]`

Status of current trial configuration.

Enumerator

NO_TEST

TEST_DELTA_C_EQ_0_DELTA_X_EQ_0

TEST_DELTA_C_GT_0_DELTA_X_EQ_0

TEST_DELTA_C_EQ_0_DELTA_X_GT_0

TEST_DELTA_C_GT_0_DELTA_X_GT_0

Definition at line 161 of file `IpCGPerturbationHandler.hpp`.

6.16.3 Constructor & Destructor Documentation

6.16.3.1 `Ipopt::CGPerturbationHandler::CGPerturbationHandler ()`

Default Constructor.

6.16.3.2 `virtual Ipopt::CGPerturbationHandler::~~CGPerturbationHandler ()` `[inline]`, `[virtual]`

Default destructor.

Definition at line 33 of file `IpCGPerturbationHandler.hpp`.

6.16.3.3 `Ipopt::CGPerturbationHandler::CGPerturbationHandler (const CGPerturbationHandler &)` `[private]`

Copy Constructor.

6.16.4 Member Function Documentation

6.16.4.1 `virtual bool Ipopt::CGPerturbationHandler::InitializeImpl (const OptionsList & options, const std::string & prefix)` `[virtual]`

Implementation of the initialization method that has to be overloaded by for each derived class.

Reimplemented from [Ipopt::PDPerturbationHandler](#).

6.16.4.2 `bool Ipopt::CGPerturbationHandler::ConsiderNewSystem (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [virtual]`

This method must be called for each new matrix, and before any other method for generating perturbation factors.

Usually, the returned perturbation factors are zero, but if the system is thought to be structurally singular, they might be positive. If the return value is false, no suitable perturbation could be found.

Reimplemented from [Ipopt::PDPerturbationHandler](#).

6.16.4.3 `bool Ipopt::CGPerturbationHandler::PerturbForSingularity (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [virtual]`

This method returns perturbation factors for the case when the most recent factorization resulted in a singular matrix.

If the return value is false, no suitable perturbation could be found.

Reimplemented from [Ipopt::PDPerturbationHandler](#).

6.16.4.4 `bool Ipopt::CGPerturbationHandler::PerturbForWrongInertia (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [virtual]`

This method returns perturbation factors for the case when the most recent factorization resulted in a matrix with an incorrect number of negative eigenvalues.

If the return value is false, no suitable perturbation could be found.

Reimplemented from [Ipopt::PDPerturbationHandler](#).

6.16.4.5 `void Ipopt::CGPerturbationHandler::CurrentPerturbation (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [virtual]`

Just return the perturbation values that have been determined most recently.

Reimplemented from [Ipopt::PDPerturbationHandler](#).

6.16.4.6 `static void Ipopt::CGPerturbationHandler::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for IpoptType.

6.16.4.7 `void Ipopt::CGPerturbationHandler::operator= (const CGPerturbationHandler &) [private]`

Overloaded Equals Operator.

6.16.4.8 `CGPenaltyData& Ipopt::CGPerturbationHandler::CGPenData () [inline],[private]`

Method to easily access CGPenalty data.

Definition at line 91 of file IpCGPerturbationHandler.hpp.

6.16.4.9 `CGPenaltyCq& Ipopt::CGPerturbationHandler::CGPenCq () [inline],[private]`

Method to easily access CGPenalty calculated quantities.

Definition at line 100 of file IpCGPerturbationHandler.hpp.

6.16.4.10 `bool Ipopt::CGPerturbationHandler::get_deltas_for_wrong_inertia (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [private]`

Internal version of PerturbForWrongInertia with the difference, that `finalize_test` is not called.

Returns false if the `delta_x` and `delta_s` parameters become too large.

6.16.4.11 `void Ipopt::CGPerturbationHandler::finalize_test () [private]`

This method is call whenever a matrix had been factorization and is not singular.

In here, we can evaluate the outcome of the deneracy test heuristics.

6.16.4.12 `Number Ipopt::CGPerturbationHandler::delta_cd () [private]`

Compute perturbation value for constraints.

6.16.5 Member Data Documentation

6.16.5.1 `Number Ipopt::CGPerturbationHandler::delta_x_last_ [private]`

The last nonzero value for `delta_x`.

Definition at line 111 of file `IpCGPerturbationHandler.hpp`.

6.16.5.2 `Number Ipopt::CGPerturbationHandler::delta_s_last_ [private]`

The last nonzero value for `delta_s`.

Definition at line 113 of file `IpCGPerturbationHandler.hpp`.

6.16.5.3 `Number Ipopt::CGPerturbationHandler::delta_c_last_ [private]`

The last nonzero value for `delta_c`.

Definition at line 115 of file `IpCGPerturbationHandler.hpp`.

6.16.5.4 `Number Ipopt::CGPerturbationHandler::delta_d_last_ [private]`

The last nonzero value for `delta_d`.

Definition at line 117 of file `IpCGPerturbationHandler.hpp`.

6.16.5.5 `Number Ipopt::CGPerturbationHandler::delta_x_curr_ [private]`

The current value for `delta_x`.

Definition at line 124 of file `IpCGPerturbationHandler.hpp`.

6.16.5.6 `Number Ipopt::CGPerturbationHandler::delta_s_curr_ [private]`

The current value for `delta_s`.

Definition at line 126 of file `IpCGPerturbationHandler.hpp`.

6.16.5.7 `Number Ipopt::CGPerturbationHandler::delta_c_curr_ [private]`

The current value for `delta_c`.

Definition at line 128 of file `IpCGPerturbationHandler.hpp`.

6.16.5.8 `Number Ipopt::CGPerturbationHandler::delta_d_curr_ [private]`

The current value for `delta_d`.

Definition at line 130 of file `IpCGPerturbationHandler.hpp`.

6.16.5.9 `bool Ipopt::CGPerturbationHandler::get_deltas_for_wrong_inertia_called_ [private]`

Flag indicating if for the given matrix the perturb for wrong inertia method has already been called.

Definition at line 135 of file IpCGPerturbationHandler.hpp.

6.16.5.10 `DegenType Ipopt::CGPerturbationHandler::hess_degenerate_ [private]`

Flag indicating whether the reduced Hessian matrix is thought to be structurally singular.

Definition at line 149 of file IpCGPerturbationHandler.hpp.

6.16.5.11 `DegenType Ipopt::CGPerturbationHandler::jac_degenerate_ [private]`

Flag indicating whether the Jacobian of the constraints is thought to be structurally rank-deficient.

Definition at line 153 of file IpCGPerturbationHandler.hpp.

6.16.5.12 `Index Ipopt::CGPerturbationHandler::degen_iters_ [private]`

Flag counting matrices in which degeneracy was observed in the first successive iterations.

-1 means that there was a non-degenerate (unperturbed) matrix at some point.

Definition at line 158 of file IpCGPerturbationHandler.hpp.

6.16.5.13 `TrialStatus Ipopt::CGPerturbationHandler::test_status_ [private]`

Current status.

Definition at line 171 of file IpCGPerturbationHandler.hpp.

6.16.5.14 `Number Ipopt::CGPerturbationHandler::delta_xs_max_ [private]`

Maximal perturbation for x and s.

Definition at line 177 of file IpCGPerturbationHandler.hpp.

6.16.5.15 `Number Ipopt::CGPerturbationHandler::delta_xs_min_ [private]`

Smallest possible perturbation for x and s.

Definition at line 179 of file IpCGPerturbationHandler.hpp.

6.16.5.16 `Number Ipopt::CGPerturbationHandler::delta_xs_first_inc_fact_ [private]`

Increase factor for delta_xs for first required perturbation.

Definition at line 181 of file IpCGPerturbationHandler.hpp.

6.16.5.17 `Number Ipopt::CGPerturbationHandler::delta_xs_inc_fact_ [private]`

Increase factor for delta_xs for later perturbations.

Definition at line 183 of file IpCGPerturbationHandler.hpp.

6.16.5.18 `Number Ipopt::CGPerturbationHandler::delta_xs_dec_fact_ [private]`

Decrease factor for delta_xs for later perturbations.

Definition at line 185 of file IpCGPerturbationHandler.hpp.

6.16.5.19 Number Ipopt::CGPerturbationHandler::delta_xs_init_ [private]

Very first trial value for delta_xs perturbation.

Definition at line 187 of file IpCGPerturbationHandler.hpp.

6.16.5.20 Number Ipopt::CGPerturbationHandler::delta_cd_val_ [private]

Size of perturbation for c and d blocks.

Definition at line 189 of file IpCGPerturbationHandler.hpp.

6.16.5.21 Number Ipopt::CGPerturbationHandler::delta_cd_exp_ [private]

Exponent on mu in formula for of perturbation for c and d blocks.

Definition at line 191 of file IpCGPerturbationHandler.hpp.

6.16.5.22 bool Ipopt::CGPerturbationHandler::reset_last_ [private]

Flag indicating whether the new values are based on the perturbations in the last iteration or in the more recent iteration in which a perturbation was done.

Definition at line 195 of file IpCGPerturbationHandler.hpp.

6.16.5.23 Index Ipopt::CGPerturbationHandler::degen_iters_max_ [private]

Required number of iterations for degeneracy conclusions.

Definition at line 197 of file IpCGPerturbationHandler.hpp.

6.16.5.24 bool Ipopt::CGPerturbationHandler::perturb_always_cd_ [private]

Flag indicating that the delta_c, delta_d perturbation should always be used.

Definition at line 200 of file IpCGPerturbationHandler.hpp.

6.16.5.25 Number Ipopt::CGPerturbationHandler::penalty_max_ [private]

The max reference value for scaling the penalty parameter.

Definition at line 204 of file IpCGPerturbationHandler.hpp.

6.16.5.26 Number Ipopt::CGPerturbationHandler::mult_diverg_feasibility_tol_ [private]

Feasibility for perturbation in pure Newton method.

Definition at line 206 of file IpCGPerturbationHandler.hpp.

The documentation for this class was generated from the following file:

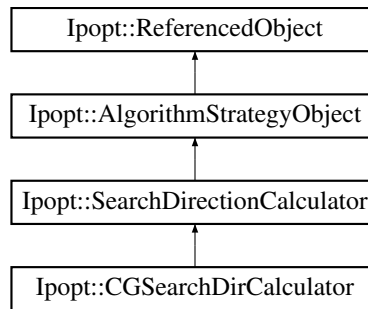
- contrib/CGPenalty/[IpCGPerturbationHandler.hpp](#)

6.17 Ipopt::CGSearchDirCalculator Class Reference

Implementation of the search direction calculator that computes the Chen-Goldfarb step for the current barrier and penalty parameter.

```
#include <IpCGSearchDirCalc.hpp>
```

Inheritance diagram for Ipopt::CGSearchDirCalculator:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [ComputeSearchDirection](#) ()
Method for computing the search direction.

Constructors/Destructors

- [CGSearchDirCalculator](#) (const [SmartPtr](#)< [PDSolver](#) > &pd_solver)
Constructor.
- virtual [~CGSearchDirCalculator](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

- [CGPenaltyData](#) & [CGPenData](#) ()
Method to easily access CGPenalty data.
- [CGPenaltyCq](#) & [CGPenCq](#) ()
Method to easily access CGPenalty calculated quantities.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CGSearchDirCalculator](#) ()
Default Constructor.
- [CGSearchDirCalculator](#) (const [CGSearchDirCalculator](#) &)
Copy Constructor.
- void [operator=](#) (const [CGSearchDirCalculator](#) &)
Overloaded Equals Operator.

Private Attributes

Algorithmic parameters

- [Number penalty_init_min_](#)
safeguard factor for bound multipliers.
- [Number penalty_init_max_](#)
Maximal value for initial penalty parameter.
- [Number penalty_max_](#)
Maximal value for penalty parameters.
- [Number pen_des_fact_](#)
parameters used in computation of line search penalty parameter and KKT perturbation parameters
- [bool penalty_backward_](#)
Algorithm type.
- [Number kappa_x_dis_](#)
parameters used to check if the fast direction can be used as the line search direction
- [Number kappa_y_dis_](#)
- [Number vartheta_](#)
- [Number delta_y_max_](#)
- [Number fast_des_fact_](#)
- [Number pen_init_fac_](#)
- [bool never_use_fact_cgpen_direction_](#)
Flag indicating whether the fast Chen-Goldfarb direction should never be used.
- [Index nonmonotone_pen_update_counter_](#)
Counter for how many times the pen para is updated nonmonotonically.

Strategy objects

- [SmartPtr< PDSystemSolver > pd_solver_](#)

Additional Inherited Members

6.17.1 Detailed Description

Implementation of the search direction calculator that computes the Chen-Goldfarb step for the current barrier and penalty parameter.

Definition at line 25 of file IpCGSearchDirCalc.hpp.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 `Ipopt::CGSearchDirCalculator::CGSearchDirCalculator (const SmartPtr< PDSystemSolver > & pd_solver)`

Constructor.

6.17.2.2 `virtual Ipopt::CGSearchDirCalculator::~~CGSearchDirCalculator () [virtual]`

Default destructor.

6.17.2.3 `Ipopt::CGSearchDirCalculator::CGSearchDirCalculator () [private]`

Default Constructor.

6.17.2.4 `Ipopt::CGSearchDirCalculator::CGSearchDirCalculator (const CGSearchDirCalculator &) [private]`

Copy Constructor.

6.17.3 Member Function Documentation

6.17.3.1 `virtual bool Ipopt::CGSearchDirCalculator::InitializImpl (const OptionsList & options, const std::string & prefix)`
[virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SearchDirectionCalculator](#).

6.17.3.2 `virtual bool Ipopt::CGSearchDirCalculator::ComputeSearchDirection ()` [virtual]

Method for computing the search direction.

If the penalty parameter has not yet been initialized, it is initialized now. The computed direction is stored in [IpData\(\).delta\(\)](#).

Implements [Ipopt::SearchDirectionCalculator](#).

6.17.3.3 `static void Ipopt::CGSearchDirCalculator::RegisterOptions (SmartPtr< RegisteredOptions > roptions)`
[static]

Methods for IpoptType.

6.17.3.4 `void Ipopt::CGSearchDirCalculator::operator= (const CGSearchDirCalculator &)` [private]

Overloaded Equals Operator.

6.17.3.5 `CGPenaltyData& Ipopt::CGSearchDirCalculator::CGPenData ()` [inline],[private]

Method to easily access CGPenalty data.

Definition at line 71 of file `IpCGSearchDirCalc.hpp`.

6.17.3.6 `CGPenaltyCq& Ipopt::CGSearchDirCalculator::CGPenCq ()` [inline],[private]

Method to easily access CGPenalty calculated quantities.

Definition at line 80 of file `IpCGSearchDirCalc.hpp`.

6.17.4 Member Data Documentation

6.17.4.1 `Number Ipopt::CGSearchDirCalculator::penalty_init_min_` [private]

safeguard factor for bound multipliers.

If value ≥ 1 , then the dual variables will never deviate from the primal estimate by more than the factors κ_{σ} and $1/\kappa_{\sigma}$.

Definition at line 94 of file `IpCGSearchDirCalc.hpp`.

6.17.4.2 `Number Ipopt::CGSearchDirCalculator::penalty_init_max_` [private]

Maximal value for initial penalty parameter.

Definition at line 96 of file `IpCGSearchDirCalc.hpp`.

6.17.4.3 `Number Ipopt::CGSearchDirCalculator::penalty_max_` [private]

Maximal value for penalty parameters.

Definition at line 98 of file IpCGSearchDirCalc.hpp.

6.17.4.4 **Number** Ipopt::CGSearchDirCalculator::pen_des_fact_ [private]

parameters used in computation of line search penalty parameter and KKT perturbation parameters

Definition at line 104 of file IpCGSearchDirCalc.hpp.

6.17.4.5 **bool** Ipopt::CGSearchDirCalculator::penalty_backward_ [private]

Algorithm type.

Definition at line 107 of file IpCGSearchDirCalc.hpp.

6.17.4.6 **Number** Ipopt::CGSearchDirCalculator::kappa_x_dis_ [private]

parameters used to check if the fast direction can be used as the line search direction

Definition at line 111 of file IpCGSearchDirCalc.hpp.

6.17.4.7 **Number** Ipopt::CGSearchDirCalculator::kappa_y_dis_ [private]

Definition at line 112 of file IpCGSearchDirCalc.hpp.

6.17.4.8 **Number** Ipopt::CGSearchDirCalculator::vartheta_ [private]

Definition at line 113 of file IpCGSearchDirCalc.hpp.

6.17.4.9 **Number** Ipopt::CGSearchDirCalculator::delta_y_max_ [private]

Definition at line 114 of file IpCGSearchDirCalc.hpp.

6.17.4.10 **Number** Ipopt::CGSearchDirCalculator::fast_des_fact_ [private]

Definition at line 115 of file IpCGSearchDirCalc.hpp.

6.17.4.11 **Number** Ipopt::CGSearchDirCalculator::pen_init_fac_ [private]

Definition at line 116 of file IpCGSearchDirCalc.hpp.

6.17.4.12 **bool** Ipopt::CGSearchDirCalculator::never_use_fact_cgpen_direction_ [private]

Flag indicating whether the fast Chen-Goldfarb direction should never be used.

Definition at line 120 of file IpCGSearchDirCalc.hpp.

6.17.4.13 **Index** Ipopt::CGSearchDirCalculator::nonmonotone_pen_update_counter_ [private]

Counter for how many times the pen para is updated nonmonotonically.

Definition at line 123 of file IpCGSearchDirCalc.hpp.

6.17.4.14 **SmartPtr<PDSysolver>** Ipopt::CGSearchDirCalculator::pd_solver_ [private]

Definition at line 128 of file IpCGSearchDirCalc.hpp.

The documentation for this class was generated from the following file:

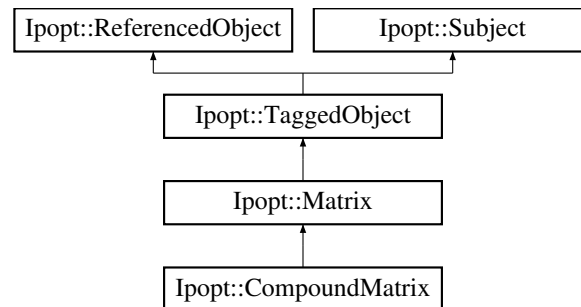
- contrib/CGPenalty/[IpCGSearchDirCalc.hpp](#)

6.18 Ipopt::CompoundMatrix Class Reference

Class for Matrices consisting of other matrices.

```
#include <IpCompoundMatrix.hpp>
```

Inheritance diagram for Ipopt::CompoundMatrix:



Public Member Functions

- void **SetComp** (Index irow, Index jcol, const **Matrix** &matrix)
Method for setting an individual component at position (irow, jcol) in the compound matrix.
- void **SetCompNonConst** (Index irow, Index jcol, **Matrix** &matrix)
*Method to set a non-const **Matrix** entry.*
- void **CreateBlockFromSpace** (Index irow, Index jcol)
Method to create a new matrix from the space for this block.
- **SmartPtr**< const **Matrix** > **GetComp** (Index irow, Index jcol) const
*Method for retrieving one block from the compound matrix as a const **Matrix**.*
- **SmartPtr**< **Matrix** > **GetCompNonConst** (Index irow, Index jcol)
*Method for retrieving one block from the compound matrix as a non-const **Matrix**.*
- Index **NComps_Rows** () const
Number of block rows of this compound matrix.
- Index **NComps_Cols** () const
Number of block columns of this compound matrix.

Constructors / Destructors

- **CompoundMatrix** (const **CompoundMatrixSpace** *owner_space)
Constructor, taking the owner_space.
- virtual **~CompoundMatrix** ()
Destructor.

Protected Member Functions

Methods overloaded from **Matrix**

- virtual void **MultVectorImpl** (Number alpha, const **Vector** &x, Number beta, **Vector** &y) const
Matrix-vector multiply.
- virtual void **TransMultVectorImpl** (Number alpha, const **Vector** &x, Number beta, **Vector** &y) const
Matrix(transpose) vector multiply.

- virtual void `AddMSinvZImpl` (Number alpha, const `Vector` &S, const `Vector` &Z, `Vector` &X) const

$$X = \text{beta} * X + \text{alpha} * (\text{Matrix } S^{-1} \{ -1 \} Z).$$
- virtual void `SinvBlrmZMTdBrlImpl` (Number alpha, const `Vector` &S, const `Vector` &R, const `Vector` &Z, const `Vector` &D, `Vector` &X) const

$$X = S^{-1} \{ -1 \} (r + \text{alpha} * Z * M^{-1} Td).$$
- virtual bool `IsValidNumbersImpl` () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void `ComputeRowAMaxImpl` (`Vector` &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void `ComputeColAMaxImpl` (`Vector` &cols_norms, bool init) const
Compute the max-norm of the columns in the matrix.
- virtual void `PrintImpl` (const `Journalist` &jnlst, `EJournalLevel` level, `EJournalCategory` category, const std::string &name, `Index` indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

- bool `MatricesValid` () const
Method to check whether or not the matrices are valid.
- const `Matrix` * `ConstComp` (`Index` irow, `Index` jcol) const
- `Matrix` * `Comp` (`Index` irow, `Index` jcol)

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `CompoundMatrix` ()
Default Constructor.
- `CompoundMatrix` (const `CompoundMatrix` &)
Copy Constructor.
- void `operator=` (const `CompoundMatrix` &)
Overloaded Equals Operator.

Private Attributes

- std::vector< std::vector
< `SmartPtr`< `Matrix` > > > `comps_`
`Matrix` of matrix's containing the components.
- std::vector< std::vector
< `SmartPtr`< const `Matrix` > > > `const_comps_`
`Matrix` of const matrix's containing the components.
- const `CompoundMatrixSpace` * `owner_space_`
Copy of the owner_space ptr as a `CompoundMatrixSpace` instead of `MatrixSpace`.
- bool `matrices_valid_`
boolean indicating if the compound matrix is in a "valid" state

Additional Inherited Members

6.18.1 Detailed Description

Class for Matrices consisting of other matrices.

This matrix is a matrix that consists of zero, one or more Matrices's which are arranged like this: $M_{\text{compound}} = \begin{pmatrix} M_{00} & M_{01} & \dots & M_{0,\text{ncomp_cols}-1} \\ \dots & & & \dots \\ M_{\text{ncomp_rows}-1,0} & M_{\text{ncomp_rows}-1,1} & \dots & M_{\text{ncomp_rows}-1,\text{ncomp_cols}-1} \end{pmatrix}$. The individual components can be associated to different MatrixSpaces. The individual components can also be const and non-const Matrices. If a component is not set (i.e., it's pointer is NULL), then this components is treated like a zero-matrix of appropriate dimensions.

Definition at line 34 of file IpCompoundMatrix.hpp.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 Ipopt::CompoundMatrix::CompoundMatrix (const CompoundMatrixSpace * owner_space)

Constructor, taking the owner_space.

The owner_space has to be defined, so that at each block row and column contain at least one non-NULL component. The individual components can be set afterwards with the SetComp and SetCompNonConst methods.

6.18.2.2 virtual Ipopt::CompoundMatrix::~~CompoundMatrix () [virtual]

Destructor.

6.18.2.3 Ipopt::CompoundMatrix::CompoundMatrix () [private]

Default Constructor.

6.18.2.4 Ipopt::CompoundMatrix::CompoundMatrix (const CompoundMatrix &) [private]

Copy Constructor.

6.18.3 Member Function Documentation

6.18.3.1 void Ipopt::CompoundMatrix::SetComp (Index irow, Index jcol, const Matrix & matrix)

Method for setting an individual component at position (irow, icol) in the compound matrix.

The counting of indices starts at 0.

6.18.3.2 void Ipopt::CompoundMatrix::SetCompNonConst (Index irow, Index jcol, Matrix & matrix)

Method to set a non-const [Matrix](#) entry.

6.18.3.3 void Ipopt::CompoundMatrix::CreateBlockFromSpace (Index irow, Index jcol)

Method to create a new matrix from the space for this block.

6.18.3.4 SmartPtr<const Matrix> Ipopt::CompoundMatrix::GetComp (Index irow, Index jcol) const [inline]

Method for retrieving one block from the compound matrix as a const [Matrix](#).

Definition at line 67 of file IpCompoundMatrix.hpp.

6.18.3.5 `SmartPointer<Matrix> Ipopt::CompoundMatrix::GetCompNonConst (Index irow, Index jcol)` `[inline]`

Method for retrieving one block from the compound matrix as a non-const [Matrix](#).

Note that calling this method with mark the [CompoundMatrix](#) as changed. Therefore, only use this method if you are intending to change the [Matrix](#) that you receive.

Definition at line 76 of file `IpCompoundMatrix.hpp`.

6.18.3.6 `Index Ipopt::CompoundMatrix::NComps_Rows () const` `[inline]`

Number of block rows of this compound matrix.

Definition at line 305 of file `IpCompoundMatrix.hpp`.

6.18.3.7 `Index Ipopt::CompoundMatrix::NComps_Cols () const` `[inline]`

Number of block columns of this compound matrix.

Definition at line 311 of file `IpCompoundMatrix.hpp`.

6.18.3.8 `virtual void Ipopt::CompoundMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const` `[protected]`, `[virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.18.3.9 `virtual void Ipopt::CompoundMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const` `[protected]`, `[virtual]`

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.18.3.10 `virtual void Ipopt::CompoundMatrix::AddMSinvZImpl (Number alpha, const Vector & S, const Vector & Z, Vector & X) const` `[protected]`, `[virtual]`

$X = \beta * X + \alpha * (\text{Matrix} S^{-1} Z)$.

Specialized implementation.

Reimplemented from [Ipopt::Matrix](#).

6.18.3.11 `virtual void Ipopt::CompoundMatrix::SinvBlrmZMTdBrlImpl (Number alpha, const Vector & S, const Vector & R, const Vector & Z, const Vector & D, Vector & X) const` `[protected]`, `[virtual]`

$X = S^{-1} (r + \alpha * Z * M^T D)$.

Specialized implementation.

Reimplemented from [Ipopt::Matrix](#).

6.18.3.12 `virtual bool Ipopt::CompoundMatrix::IsValidNumbersImpl () const` `[protected]`, `[virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.18.3.13 `virtual void Ipopt::CompoundMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const`
`[protected], [virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.18.3.14 `virtual void Ipopt::CompoundMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const`
`[protected], [virtual]`

Compute the max-norm of the columns in the matrix.

The result is stored in cols_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.18.3.15 `virtual void Ipopt::CompoundMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` `[protected], [virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.18.3.16 `void Ipopt::CompoundMatrix::operator= (const CompoundMatrix &)` `[private]`

Overloaded Equals Operator.

6.18.3.17 `bool Ipopt::CompoundMatrix::MatricesValid () const` `[private]`

Method to check whether or not the matrices are valid.

6.18.3.18 `const Matrix * Ipopt::CompoundMatrix::ConstComp (Index irow, Index jcol) const` `[inline], [private]`

Definition at line 317 of file IpCompoundMatrix.hpp.

6.18.3.19 `Matrix * Ipopt::CompoundMatrix::Comp (Index irow, Index jcol)` `[inline], [private]`

Definition at line 332 of file IpCompoundMatrix.hpp.

6.18.4 Member Data Documentation

6.18.4.1 `std::vector<std::vector<SmartPointer<Matrix>>> Ipopt::CompoundMatrix::comps_` `[private]`

[Matrix](#) of matrix's containing the components.

Definition at line 143 of file IpCompoundMatrix.hpp.

6.18.4.2 `std::vector<std::vector<SmartPointer<const Matrix>>> Ipopt::CompoundMatrix::const_comps_` `[private]`

[Matrix](#) of const matrix's containing the components.

Definition at line 146 of file IpCompoundMatrix.hpp.

6.18.4.3 `const CompoundMatrixSpace* Ipopt::CompoundMatrix::owner_space_` `[private]`

Copy of the owner_space ptr as a [CompoundMatrixSpace](#) instead of [MatrixSpace](#).

Definition at line 150 of file IpCompoundMatrix.hpp.

6.18.4.4 `bool Ipopt::CompoundMatrix::matrices_valid_` [mutable], [private]

boolean indicating if the compound matrix is in a "valid" state

Definition at line 153 of file `IpCompoundMatrix.hpp`.

The documentation for this class was generated from the following file:

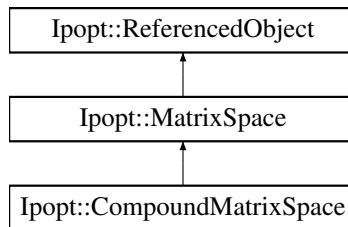
- [LinAlg/IpCompoundMatrix.hpp](#)

6.19 `Ipopt::CompoundMatrixSpace` Class Reference

This is the matrix space for [CompoundMatrix](#).

```
#include <IpCompoundMatrix.hpp>
```

Inheritance diagram for `Ipopt::CompoundMatrixSpace`:



Public Member Functions

- `SmartPointer< const MatrixSpace > GetCompSpace (Index irow, Index jcol) const`
Obtain the component [MatrixSpace](#) in block row *irow* and block column *jcol*.
- `CompoundMatrix * MakeNewCompoundMatrix () const`
Method for creating a new matrix of this specific type.
- `virtual Matrix * MakeNew () const`
Overloaded `MakeNew` method for the [MatrixSpace](#) base class.

Constructors / Destructors

- `CompoundMatrixSpace (Index ncomps_rows, Index ncomps_cols, Index total_nRows, Index total_nCols)`
Constructor, given the number of row and columns blocks, as well as the total number of rows and columns.
- `~CompoundMatrixSpace ()`
Destructor.

Methods for setting information about the components.

- `void SetBlockRows (Index irow, Index nrows)`
Set the number *nrows* of rows in row-block number *irow*.
- `void SetBlockCols (Index jcol, Index ncols)`
Set the number *ncols* of columns in column-block number *jcol*.
- `Index GetBlockRows (Index irow) const`
Get the number *nrows* of rows in row-block number *irow*.
- `Index GetBlockCols (Index jcol) const`
Set the number *ncols* of columns in column-block number *jcol*.
- `void SetCompSpace (Index irow, Index jcol, const MatrixSpace &mat_space, bool auto_allocate=false)`

Set the component [MatrixSpace](#).

Accessor methods

- [Index NComps_Rows](#) () const
Number of block rows.
- [Index NComps_Cols](#) () const
Number of block columns.
- bool [Diagonal](#) () const
True if the blocks lie on the diagonal - can make some operations faster.

Private Member Functions

- bool [DimensionsSet](#) () const
Auxilliary function for debugging to set if all block dimensions have been set.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CompoundMatrixSpace](#) ()
Default constructor.
- [CompoundMatrixSpace](#) (const [CompoundMatrixSpace](#) &)
Copy Constructor.
- [CompoundMatrixSpace](#) & operator= (const [CompoundMatrixSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- [Index ncomps_rows_](#)
Number of block rows.
- [Index ncomps_cols_](#)
Number of block columns.
- bool [dimensions_set_](#)
Store whether or not the dimensions are valid.
- std::vector< std::vector
< [SmartPtr](#)< const [MatrixSpace](#) > > > [comp_spaces_](#)
2-dim std::vector of matrix spaces for the components
- std::vector< std::vector< bool > > [allocate_block_](#)
2-dim std::vector of booleans deciding whether to allocate a new matrix for the blocks automagically
- std::vector< [Index](#) > [block_rows_](#)
Vector of the number of rows in each comp column.
- std::vector< [Index](#) > [block_cols_](#)
Vector of the number of cols in each comp row.
- bool [diagonal_](#)
true if the [CompoundMatrixSpace](#) only has [Matrix](#) spaces along the diagonal.

6.19.1 Detailed Description

This is the matrix space for [CompoundMatrix](#).

Before a [CompoundMatrix](#) can be created, at least one [MatrixSpace](#) has to be set per block row and column. Individual component [MatrixSpace](#)'s can be set with the SetComp method.

Definition at line 168 of file IpCompoundMatrix.hpp.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 `Ipopt::CompoundMatrixSpace::CompoundMatrixSpace (Index ncomps_rows, Index ncomps_cols, Index total_nRows, Index total_nCols)`

Constructor, given the number of row and columns blocks, as well as the total number of rows and columns.

6.19.2.2 `Ipopt::CompoundMatrixSpace::~~CompoundMatrixSpace () [inline]`

Destructor.

Definition at line 182 of file IpCompoundMatrix.hpp.

6.19.2.3 `Ipopt::CompoundMatrixSpace::CompoundMatrixSpace () [private]`

Default constructor.

6.19.2.4 `Ipopt::CompoundMatrixSpace::CompoundMatrixSpace (const CompoundMatrixSpace &) [private]`

Copy Constructor.

6.19.3 Member Function Documentation

6.19.3.1 `void Ipopt::CompoundMatrixSpace::SetBlockRows (Index irow, Index nRows)`

Set the number nRows of rows in row-block number irow.

6.19.3.2 `void Ipopt::CompoundMatrixSpace::SetBlockCols (Index jcol, Index nCols)`

Set the number nCols of columns in column-block number jcol.

6.19.3.3 `Index Ipopt::CompoundMatrixSpace::GetBlockRows (Index irow) const`

Get the number nRows of rows in row-block number irow.

6.19.3.4 `Index Ipopt::CompoundMatrixSpace::GetBlockCols (Index jcol) const`

Set the number nCols of columns in column-block number jcol.

6.19.3.5 `void Ipopt::CompoundMatrixSpace::SetCompSpace (Index irow, Index jcol, const MatrixSpace & mat_space, bool auto_allocate = false)`

Set the component [MatrixSpace](#).

If auto_allocate is true, then a new [CompoundMatrix](#) created later with MakeNew will have this component automatically created with the [Matrix](#)'s MakeNew. Otherwise, the corresponding component will be NULL and has to be set with the SetComp methods of the [CompoundMatrix](#).

6.19.3.6 `SmartPointer<const MatrixSpace> Ipopt::CompoundMatrixSpace::GetCompSpace (Index irow, Index jcol) const`
`[inline]`

Obtain the component [MatrixSpace](#) in block row *irow* and block column *jcol*.

Definition at line 214 of file `IpCompoundMatrix.hpp`.

6.19.3.7 `Index Ipopt::CompoundMatrixSpace::NComps_Rows () const` `[inline]`

Number of block rows.

Definition at line 224 of file `IpCompoundMatrix.hpp`.

6.19.3.8 `Index Ipopt::CompoundMatrixSpace::NComps_Cols () const` `[inline]`

Number of block columns.

Definition at line 229 of file `IpCompoundMatrix.hpp`.

6.19.3.9 `bool Ipopt::CompoundMatrixSpace::Diagonal () const` `[inline]`

True if the blocks lie on the diagonal - can make some operations faster.

Definition at line 235 of file `IpCompoundMatrix.hpp`.

6.19.3.10 `CompoundMatrix* Ipopt::CompoundMatrixSpace::MakeNewCompoundMatrix () const`

Method for creating a new matrix of this specific type.

6.19.3.11 `virtual Matrix* Ipopt::CompoundMatrixSpace::MakeNew () const` `[inline], [virtual]`

Overloaded `MakeNew` method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 246 of file `IpCompoundMatrix.hpp`.

6.19.3.12 `CompoundMatrixSpace& Ipopt::CompoundMatrixSpace::operator= (const CompoundMatrixSpace &)`
`[private]`

Overloaded Equals Operator.

6.19.3.13 `bool Ipopt::CompoundMatrixSpace::DimensionsSet () const` `[private]`

Auxilliary function for debugging to set if all block dimensions have been set.

6.19.4 Member Data Documentation

6.19.4.1 `Index Ipopt::CompoundMatrixSpace::ncomps_rows_` `[private]`

Number of block rows.

Definition at line 271 of file `IpCompoundMatrix.hpp`.

6.19.4.2 `Index Ipopt::CompoundMatrixSpace::ncomps_cols_` `[private]`

Number of block columns.

Definition at line 274 of file `IpCompoundMatrix.hpp`.

6.19.4.3 `bool Ipopt::CompoundMatrixSpace::dimensions_set_ [mutable], [private]`

Store whether or not the dimensions are valid.

Definition at line 277 of file `IpCompoundMatrix.hpp`.

6.19.4.4 `std::vector<std::vector<SmartPointer<const MatrixSpace>>> Ipopt::CompoundMatrixSpace::comp_spaces_ [private]`

2-dim `std::vector` of matrix spaces for the components

Definition at line 280 of file `IpCompoundMatrix.hpp`.

6.19.4.5 `std::vector<std::vector< bool >> Ipopt::CompoundMatrixSpace::allocate_block_ [private]`

2-dim `std::vector` of booleans deciding whether to allocate a new matrix for the blocks automatically

Definition at line 284 of file `IpCompoundMatrix.hpp`.

6.19.4.6 `std::vector<Index> Ipopt::CompoundMatrixSpace::block_rows_ [private]`

[Vector](#) of the number of rows in each comp column.

Definition at line 287 of file `IpCompoundMatrix.hpp`.

6.19.4.7 `std::vector<Index> Ipopt::CompoundMatrixSpace::block_cols_ [private]`

[Vector](#) of the number of cols in each comp row.

Definition at line 290 of file `IpCompoundMatrix.hpp`.

6.19.4.8 `bool Ipopt::CompoundMatrixSpace::diagonal_ [private]`

true if the [CompoundMatrixSpace](#) only has [Matrix](#) spaces along the diagonal.

this means that the [CompoundMatrix](#) will only have matrices along the diagonal and it could make some operations more efficient

Definition at line 296 of file `IpCompoundMatrix.hpp`.

The documentation for this class was generated from the following file:

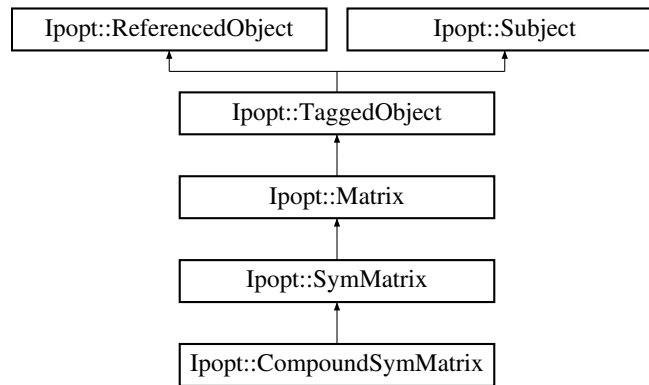
- [LinAlg/IpCompoundMatrix.hpp](#)

6.20 Ipopt::CompoundSymMatrix Class Reference

Class for symmetric matrices consisting of other matrices.

```
#include <IpCompoundSymMatrix.hpp>
```

Inheritance diagram for `Ipopt::CompoundSymMatrix`:



Public Member Functions

- void **SetComp** (Index irow, Index jcol, const Matrix &matrix)
Method for setting an individual component at position (irow, jcol) in the compound matrix.
- void **SetCompNonConst** (Index irow, Index jcol, Matrix &matrix)
Non const version of the same method.
- SmartPtr< const Matrix > **GetComp** (Index irow, Index jcol) const
Method for retrieving one block from the compound matrix.
- SmartPtr< Matrix > **GetCompNonConst** (Index irow, Index jcol)
Non const version of GetComp.
- SmartPtr< CompoundSymMatrix > **MakeNewCompoundSymMatrix** () const
Method for creating a new matrix of this specific type.
- Index **NComps_Dim** () const
Number of block rows and columns.

Constructors / Destructors

- **CompoundSymMatrix** (const CompoundSymMatrixSpace *owner_space)
Constructor, taking only the number for block components into the row and column direction.
- **~CompoundSymMatrix** ()
Destructor.

Protected Member Functions

Methods overloaded from matrix

- virtual void **MultVectorImpl** (Number alpha, const Vector &x, Number beta, Vector &y) const
Matrix-vector multiply.
- virtual bool **IsValidNumbersImpl** () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void **ComputeRowAMaxImpl** (Vector &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void **PrintImpl** (const Journalist &jnlst, EJournalLevel level, EJournalCategory category, const std::string &name, Index indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

- `bool MatricesValid () const`
method to check whether or not the matrices are valid
- `const Matrix * ConstComp (Index irow, Index jcol) const`
Internal method to return a const pointer to one of the comps.
- `Matrix * Comp (Index irow, Index jcol)`
Internal method to return a non-const pointer to one of the comps.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `CompoundSymMatrix ()`
Default Constructor.
- `CompoundSymMatrix (const CompoundSymMatrix &)`
Copy Constructor.
- `void operator= (const CompoundSymMatrix &)`
Overloaded Equals Operator.

Private Attributes

- `std::vector< std::vector
< SmartPtr< Matrix > > > comps_`
Vector of vectors containing the components.
- `std::vector< std::vector
< SmartPtr< const Matrix > > > const_comps_`
Vector of vectors containing the const components.
- `const CompoundSymMatrixSpace * owner_space_`
Copy of the owner_space ptr as a CompoundSymMatrixSpace.
- `bool matrices_valid_`
boolean indicating if the compound matrix is in a "valid" state

Additional Inherited Members

6.20.1 Detailed Description

Class for symmetric matrices consisting of other matrices.

Here, the lower left block of the matrix is stored.

Definition at line 24 of file `IpCompoundSymMatrix.hpp`.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 `Ipopt::CompoundSymMatrix::CompoundSymMatrix (const CompoundSymMatrixSpace * owner_space)`

Constructor, taking only the number for block components into the row and column direction.

The `owner_space` has to be defined, so that at each block row and column contain at least one non-NULL component.

6.20.2.2 Ipopt::CompoundSymMatrix::~~CompoundSymMatrix ()

Destructor.

6.20.2.3 Ipopt::CompoundSymMatrix::CompoundSymMatrix () [private]

Default Constructor.

6.20.2.4 Ipopt::CompoundSymMatrix::CompoundSymMatrix (const CompoundSymMatrix &) [private]

Copy Constructor.

6.20.3 Member Function Documentation

6.20.3.1 void Ipopt::CompoundSymMatrix::SetComp (Index *irow*, Index *jcol*, const Matrix & *matrix*)

Method for setting an individual component at position (*irow*, *jcol*) in the compound matrix.

The counting of indices starts at 0. Since this only the lower left components are stored, we need to have $jcol \leq irow$, and if $irow == jcol$, the matrix must be a [SymMatrix](#)

6.20.3.2 void Ipopt::CompoundSymMatrix::SetCompNonConst (Index *irow*, Index *jcol*, Matrix & *matrix*)

Non const version of the same method.

6.20.3.3 SmartPtr<const Matrix> Ipopt::CompoundSymMatrix::GetComp (Index *irow*, Index *jcol*) const [inline]

Method for retrieving one block from the compound matrix.

Since this only the lower left components are stored, we need to have $jcol \leq irow$

Definition at line 54 of file IpCompoundSymMatrix.hpp.

6.20.3.4 SmartPtr<Matrix> Ipopt::CompoundSymMatrix::GetCompNonConst (Index *irow*, Index *jcol*) [inline]

Non const version of GetComp.

You should only use this method if you are intending to change the matrix you receive, since this [CompoundSymMatrix](#) will be marked as changed.

Definition at line 62 of file IpCompoundSymMatrix.hpp.

6.20.3.5 SmartPtr<CompoundSymMatrix> Ipopt::CompoundSymMatrix::MakeNewCompoundSymMatrix () const [inline]

Method for creating a new matrix of this specific type.

Definition at line 277 of file IpCompoundSymMatrix.hpp.

6.20.3.6 Index Ipopt::CompoundSymMatrix::NComps_Dim () const

Number of block rows and columns.

6.20.3.7 virtual void Ipopt::CompoundSymMatrix::MultVectorImpl (Number *alpha*, const Vector & *x*, Number *beta*, Vector & *y*) const [protected], [virtual]

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [lpopt::Matrix](#).

6.20.3.8 `virtual bool lpopt::CompoundSymMatrix::IsValidNumbersImpl () const [protected],[virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [lpopt::Matrix](#).

6.20.3.9 `virtual void lpopt::CompoundSymMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const [protected],[virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [lpopt::Matrix](#).

6.20.3.10 `virtual void lpopt::CompoundSymMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const [protected],[virtual]`

Print detailed information about the matrix.

Implements [lpopt::Matrix](#).

6.20.3.11 `void lpopt::CompoundSymMatrix::operator= (const CompoundSymMatrix &) [private]`

Overloaded Equals Operator.

6.20.3.12 `bool lpopt::CompoundSymMatrix::MatricesValid () const [private]`

method to check whether or not the matrices are valid

6.20.3.13 `const Matrix* lpopt::CompoundSymMatrix::ConstComp (Index irow, Index jcol) const [inline],[private]`

Internal method to return a const pointer to one of the comps.

Definition at line 136 of file lpCompoundSymMatrix.hpp.

6.20.3.14 `Matrix* lpopt::CompoundSymMatrix::Comp (Index irow, Index jcol) [inline],[private]`

Internal method to return a non-const pointer to one of the comps.

Definition at line 151 of file lpCompoundSymMatrix.hpp.

6.20.4 Member Data Documentation

6.20.4.1 `std::vector<std::vector<SmartPtr<Matrix>>> lpopt::CompoundSymMatrix::comps_ [private]`

[Vector](#) of vectors containing the components.

Definition at line 121 of file lpCompoundSymMatrix.hpp.

6.20.4.2 `std::vector<std::vector<SmartPtr<const Matrix>>> lpopt::CompoundSymMatrix::const_comps_ [private]`

[Vector](#) of vectors containing the const components.

Definition at line 124 of file lpCompoundSymMatrix.hpp.

6.20.4.3 `const CompoundSymMatrixSpace* Ipopt::CompoundSymMatrix::owner_space_ [private]`

Copy of the owner_space ptr as a [CompoundSymMatrixSpace](#).

Definition at line 127 of file `IpCompoundSymMatrix.hpp`.

6.20.4.4 `bool Ipopt::CompoundSymMatrix::matrices_valid_ [mutable], [private]`

boolean indicating if the compound matrix is in a "valid" state

Definition at line 130 of file `IpCompoundSymMatrix.hpp`.

The documentation for this class was generated from the following file:

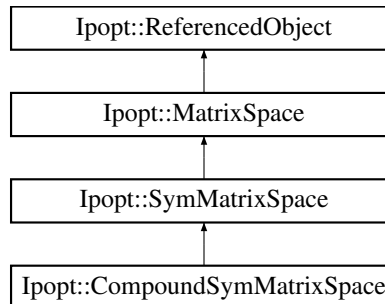
- [LinAlg/IpCompoundSymMatrix.hpp](#)

6.21 Ipopt::CompoundSymMatrixSpace Class Reference

This is the matrix space for [CompoundSymMatrix](#).

```
#include <IpCompoundSymMatrix.hpp>
```

Inheritance diagram for `Ipopt::CompoundSymMatrixSpace`:



Public Member Functions

- `SmartPtr< const MatrixSpace > GetCompSpace (Index irow, Index jcol) const`
Obtain the component [MatrixSpace](#) in block row irow and block column jcol.
- `CompoundSymMatrix * MakeNewCompoundSymMatrix () const`
Method for creating a new matrix of this specific type.
- `virtual SymMatrix * MakeNewSymMatrix () const`
Overloaded MakeNew method for the [SymMatrixSpace](#) base class.

Constructors / Destructors

- `CompoundSymMatrixSpace (Index ncomp_spaces, Index total_dim)`
Constructor, given the number of blocks (same for rows and columns), as well as the total dimension of the matrix.
- `~CompoundSymMatrixSpace ()`
Destructor.

Methods for setting information about the components.

- `void SetBlockDim (Index irow_jcol, Index dim)`

- Set the dimension *dim* for block row (or column) *irow_jcol*.
- [Index GetBlockDim](#) ([Index irow_jcol](#)) const
Get the dimension *dim* for block row (or column) *irow_jcol*.
- void [SetCompSpace](#) ([Index irow](#), [Index jcol](#), const [MatrixSpace](#) &mat_space, bool auto_allocate=false)
Set the component [SymMatrixSpace](#).

Accessor methods

- [Index NComps_Dim](#) () const

Private Member Functions

- bool [DimensionsSet](#) () const
Method to check whether or not the spaces are valid.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CompoundSymMatrixSpace](#) ()
Default constructor.
- [CompoundSymMatrixSpace](#) (const [CompoundSymMatrix](#) &)
Copy Constructor.
- [CompoundSymMatrixSpace](#) & operator= (const [CompoundSymMatrixSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- [Index ncomp_spaces_](#)
Number of components per row and column.
- std::vector< [Index](#) > [block_dim_](#)
[Vector](#) of the number of rows in each comp column, Since this is symmetric, this is also the number of columns in each row, hence, it is the dimension each of the diagonals.
- std::vector< std::vector
< [SmartPtr](#)< const [MatrixSpace](#) > > > [comp_spaces_](#)
2-dim std::vector of matrix spaces for the components.
- std::vector< std::vector< bool > > [allocate_block_](#)
2-dim std::vector of booleans deciding whether to allocate a new matrix for the blocks automatically
- bool [dimensions_set_](#)
boolean indicating if the compound matrix space is in a "valid" state

6.21.1 Detailed Description

This is the matrix space for [CompoundSymMatrix](#).

Before a [CompoundSymMatrix](#) can be created, at least one [SymMatrixSpace](#) has to be set per block row and column. Individual component [SymMatrixSpace](#)'s can be set with the SetComp method.

Definition at line 171 of file [IpCompoundSymMatrix.hpp](#).

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Ipopt::CompoundSymMatrixSpace::CompoundSymMatrixSpace (Index ncomp_spaces, Index total_dim)

Constructor, given the number of blocks (same for rows and columns), as well as the total dimension of the matrix.

6.21.2.2 Ipopt::CompoundSymMatrixSpace::~~CompoundSymMatrixSpace () [inline]

Destructor.

Definition at line 182 of file IpCompoundSymMatrix.hpp.

6.21.2.3 Ipopt::CompoundSymMatrixSpace::CompoundSymMatrixSpace () [private]

Default constructor.

6.21.2.4 Ipopt::CompoundSymMatrixSpace::CompoundSymMatrixSpace (const CompoundSymMatrix &) [private]

Copy Constructor.

6.21.3 Member Function Documentation

6.21.3.1 void Ipopt::CompoundSymMatrixSpace::SetBlockDim (Index irow_jcol, Index dim)

Set the dimension dim for block row (or column) irow_jcol.

6.21.3.2 Index Ipopt::CompoundSymMatrixSpace::GetBlockDim (Index irow_jcol) const

Get the dimension dim for block row (or column) irow_jcol.

6.21.3.3 void Ipopt::CompoundSymMatrixSpace::SetCompSpace (Index irow, Index jcol, const MatrixSpace & mat_space, bool auto_allocate = false)

Set the component [SymMatrixSpace](#).

If auto_allocate is true, then a new [CompoundSymMatrix](#) created later with MakeNew will have this component automatically created with the [SymMatrix](#)'s MakeNew. Otherwise, the corresponding component will be NULL and has to be set with the SetComp methods of the [CompoundSymMatrix](#).

6.21.3.4 SmartPtr<const MatrixSpace> Ipopt::CompoundSymMatrixSpace::GetCompSpace (Index irow, Index jcol) const [inline]

Obtain the component [MatrixSpace](#) in block row irow and block column jcol.

Definition at line 208 of file IpCompoundSymMatrix.hpp.

6.21.3.5 Index Ipopt::CompoundSymMatrixSpace::NComps_Dim () const [inline]

Definition at line 217 of file IpCompoundSymMatrix.hpp.

6.21.3.6 CompoundSymMatrix* Ipopt::CompoundSymMatrixSpace::MakeNewCompoundSymMatrix () const

Method for creating a new matrix of this specific type.

6.21.3.7 virtual SymMatrix* Ipopt::CompoundSymMatrixSpace::MakeNewSymMatrix () const [inline],[virtual]

Overloaded MakeNew method for the [SymMatrixSpace](#) base class.

Implements [Ipopt::SymMatrixSpace](#).

Definition at line 228 of file IpCompoundSymMatrix.hpp.

6.21.3.8 CompoundSymMatrixSpace& Ipopt::CompoundSymMatrixSpace::operator= (const CompoundSymMatrixSpace &) [private]

Overloaded Equals Operator.

6.21.3.9 bool Ipopt::CompoundSymMatrixSpace::DimensionsSet () const [private]

Method to check whether or not the spaces are valid.

6.21.4 Member Data Documentation

6.21.4.1 Index Ipopt::CompoundSymMatrixSpace::ncomp_spaces_ [private]

Number of components per row and column.

Definition at line 253 of file IpCompoundSymMatrix.hpp.

6.21.4.2 std::vector<Index> Ipopt::CompoundSymMatrixSpace::block_dim_ [private]

[Vector](#) of the number of rows in each comp column, Since this is symmetric, this is also the number of columns in each row, hence, it is the dimension each of the diagonals.

Definition at line 259 of file IpCompoundSymMatrix.hpp.

6.21.4.3 std::vector<std::vector<SmartPtr<const MatrixSpace> > > Ipopt::CompoundSymMatrixSpace::comp_spaces_ [private]

2-dim std::vector of matrix spaces for the components.

Only the lower right part is stored.

Definition at line 263 of file IpCompoundSymMatrix.hpp.

6.21.4.4 std::vector<std::vector< bool > > Ipopt::CompoundSymMatrixSpace::allocate_block_ [private]

2-dim std::vector of booleans deciding whether to allocate a new matrix for the blocks automagically

Definition at line 267 of file IpCompoundSymMatrix.hpp.

6.21.4.5 bool Ipopt::CompoundSymMatrixSpace::dimensions_set_ [mutable],[private]

boolean indicating if the compound matrix space is in a "valid" state

Definition at line 270 of file IpCompoundSymMatrix.hpp.

The documentation for this class was generated from the following file:

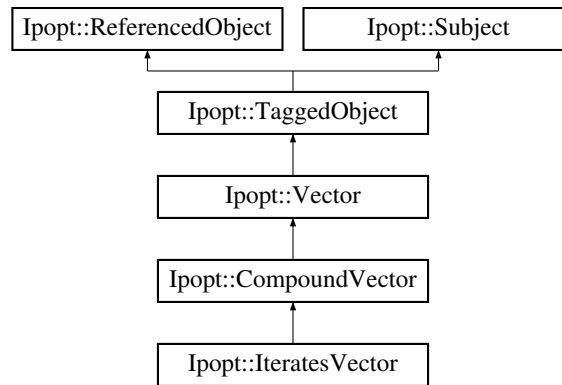
- [LinAlg/IpCompoundSymMatrix.hpp](#)

6.22 Ipopt::CompoundVector Class Reference

Class of Vectors consisting of other vectors.

```
#include <IpCompoundVector.hpp>
```

Inheritance diagram for Ipopt::CompoundVector:



Public Member Functions

- void **SetComp** (Index icomp, const **Vector** &vec)
*Method for setting the pointer for a component that is a const **Vector**.*
- void **SetCompNonConst** (Index icomp, **Vector** &vec)
*Method for setting the pointer for a component that is a non-const **Vector**.*
- Index **NComps** () const
Number of components of this compound vector.
- bool **IsCompConst** (Index i) const
Check if a particular component is const or not.
- bool **IsCompNull** (Index i) const
Check if a particular component is null or not.
- SmartPtr< const **Vector** > **GetComp** (Index i) const
Return a particular component (const version)
- SmartPtr< **Vector** > **GetCompNonConst** (Index i)
Return a particular component (non-const version).

Constructors/Destructors

- **CompoundVector** (const **CompoundVectorSpace** *owner_space, bool create_new)
*Constructor, given the corresponding **CompoundVectorSpace**.*
- virtual **~CompoundVector** ()
Default destructor.

Protected Member Functions

- virtual bool **IsValidNumbersImpl** () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Overloaded methods from Vector base class

- virtual void **CopyImpl** (const **Vector** &x)
Copy the data of the vector x into this vector (DCOPY).
- virtual void **ScallImpl** (Number alpha)
Scales the vector by scalar alpha (DSCAL)
- virtual void **AxpyImpl** (Number alpha, const **Vector** &x)
Add the multiple alpha of vector x to this vector (DAXPY)

- virtual `Number DotImpl` (const `Vector` &x) const
Computes inner product of vector x with this (DDOT)
- virtual `Number Nrm2Impl` () const
Computes the 2-norm of this vector (DNRM2)
- virtual `Number AsumImpl` () const
Computes the 1-norm of this vector (DASUM)
- virtual `Number AmaxImpl` () const
Computes the max-norm of this vector (based on IDAMAX)
- virtual void `SetImpl` (`Number` value)
Set each element in the vector to the scalar alpha.
- virtual void `ElementWiseDivideImpl` (const `Vector` &x)
Element-wise division $y_i \leftarrow y_i / x_i$.
- virtual void `ElementWiseMultiplyImpl` (const `Vector` &x)
*Element-wise multiplication $y_i \leftarrow y_i * x_i$.*
- virtual void `ElementWiseMaxImpl` (const `Vector` &x)
Element-wise max against entries in x.
- virtual void `ElementWiseMinImpl` (const `Vector` &x)
Element-wise min against entries in x.
- virtual void `ElementWiseReciprocalImpl` ()
Element-wise reciprocal.
- virtual void `ElementWiseAbsImpl` ()
Element-wise absolute values.
- virtual void `ElementWiseSqrtImpl` ()
Element-wise square-root.
- virtual void `ElementWiseSgnImpl` ()
Replaces entries with sgn of the entry.
- virtual void `AddScalarImpl` (`Number` scalar)
Add scalar to every component of the vector.
- virtual `Number MaxImpl` () const
Max value in the vector.
- virtual `Number MinImpl` () const
Min value in the vector.
- virtual `Number SumImpl` () const
Computes the sum of the elements of vector.
- virtual `Number SumLogsImpl` () const
Computes the sum of the logs of the elements of vector.

Implemented specialized functions

- void `AddTwoVectorsImpl` (`Number` a, const `Vector` &v1, `Number` b, const `Vector` &v2, `Number` c)
*Add two vectors ($a * v1 + b * v2$).*
- `Number FracToBoundImpl` (const `Vector` &delta, `Number` tau) const
Fraction to the boundary parameter.
- void `AddVectorQuotientImpl` (`Number` a, const `Vector` &z, const `Vector` &s, `Number` c)
*Add the quotient of two vectors, $y = a * z/s + c * y$.*

Output methods

- virtual void `PrintImpl` (const `Journalist` &jnlst, `EJournalLevel` level, `EJournalCategory` category, const std::string &name, `Index` indent, const std::string &prefix) const
Print the entire vector.

Private Member Functions

- bool [VectorsValid](#) ()
- const [Vector](#) * [ConstComp](#) ([Index](#) i) const
- [Vector](#) * [Comp](#) ([Index](#) i)

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CompoundVector](#) ()
Default Constructor.
- [CompoundVector](#) (const [CompoundVector](#) &)
Copy Constructor.
- void [operator=](#) (const [CompoundVector](#) &)
Overloaded Equals Operator.

Private Attributes

- std::vector< [SmartPtr](#)< [Vector](#) > > [comps_](#)
Components of the compound vector.
- std::vector< [SmartPtr](#)< const [Vector](#) > > [const_comps_](#)
- const [CompoundVectorSpace](#) * [owner_space_](#)
- bool [vectors_valid_](#)

Additional Inherited Members

6.22.1 Detailed Description

Class of Vectors consisting of other vectors.

This vector is a vector that consists of zero, one or more [Vector](#)'s which are stacked on each others: $x_{\text{compound}} =$

$\begin{pmatrix} x_0 \\ \dots \\ x_{\text{ncomps}-1} \end{pmatrix}$. The individual components can be associated to different VectorSpaces. The individual components can also be const and non-const Vectors.

Definition at line 30 of file IpCompoundVector.hpp.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Ipopt::CompoundVector::CompoundVector (const [CompoundVectorSpace](#) * [owner_space](#), bool [create_new](#))

Constructor, given the corresponding [CompoundVectorSpace](#).

Before this constructor can be called, all components of the [CompoundVectorSpace](#) have to be set, so that the constructors for the individual components can be called. If the flag [create_new](#) is true, then the individual components of the new [CompoundVector](#) are initialized with the MakeNew methods of each [VectorSpace](#) (and are non-const). Otherwise, the individual components can later be set using the SetComp and SetCompNonConst method.

6.22.2.2 `virtual Ipopt::CompoundVector::~~CompoundVector () [virtual]`

Default destructor.

6.22.2.3 `Ipopt::CompoundVector::CompoundVector () [private]`

Default Constructor.

6.22.2.4 `Ipopt::CompoundVector::CompoundVector (const CompoundVector &) [private]`

Copy Constructor.

6.22.3 Member Function Documentation

6.22.3.1 `void Ipopt::CompoundVector::SetComp (Index icomp, const Vector & vec)`

Method for setting the pointer for a component that is a const [Vector](#).

6.22.3.2 `void Ipopt::CompoundVector::SetCompNonConst (Index icomp, Vector & vec)`

Method for setting the pointer for a component that is a non-const [Vector](#).

6.22.3.3 `Index Ipopt::CompoundVector::NComps () const [inline]`

Number of components of this compound vector.

Definition at line 308 of file `IpCompoundVector.hpp`.

6.22.3.4 `bool Ipopt::CompoundVector::IsCompConst (Index i) const [inline]`

Check if a particular component is const or not.

Definition at line 65 of file `IpCompoundVector.hpp`.

6.22.3.5 `bool Ipopt::CompoundVector::IsCompNull (Index i) const [inline]`

Check if a particular component is null or not.

Definition at line 76 of file `IpCompoundVector.hpp`.

6.22.3.6 `SmartPointer<const Vector> Ipopt::CompoundVector::GetComp (Index i) const [inline]`

Return a particular component (const version)

Definition at line 86 of file `IpCompoundVector.hpp`.

6.22.3.7 `SmartPointer<Vector> Ipopt::CompoundVector::GetCompNonConst (Index i) [inline]`

Return a particular component (non-const version).

Note that calling this method with mark the [CompoundVector](#) as changed. Therefore, only use this method if you are intending to change the [Vector](#) that you receive.

Definition at line 96 of file `IpCompoundVector.hpp`.

6.22.3.8 `virtual void Ipopt::CompoundVector::CopyImpl (const Vector & x) [protected],[virtual]`

Copy the data of the vector *x* into this vector (DCOPY).

Implements [Ipopt::Vector](#).

6.22.3.9 `virtual void Ipopt::CompoundVector::ScallImpl (Number alpha)` [protected],[virtual]

Scales the vector by scalar alpha (DSCAL)

Implements [Ipopt::Vector](#).

6.22.3.10 `virtual void Ipopt::CompoundVector::AxpymImpl (Number alpha, const Vector & x)` [protected],[virtual]

Add the multiple alpha of vector x to this vector (DAXPY)

Implements [Ipopt::Vector](#).

6.22.3.11 `virtual Number Ipopt::CompoundVector::DotImpl (const Vector & x) const` [protected],[virtual]

Computes inner product of vector x with this (DDOT)

Implements [Ipopt::Vector](#).

6.22.3.12 `virtual Number Ipopt::CompoundVector::Nrm2Impl () const` [protected],[virtual]

Computes the 2-norm of this vector (DNRM2)

Implements [Ipopt::Vector](#).

6.22.3.13 `virtual Number Ipopt::CompoundVector::AsumImpl () const` [protected],[virtual]

Computes the 1-norm of this vector (DASUM)

Implements [Ipopt::Vector](#).

6.22.3.14 `virtual Number Ipopt::CompoundVector::AmaxImpl () const` [protected],[virtual]

Computes the max-norm of this vector (based on IDAMAX)

Implements [Ipopt::Vector](#).

6.22.3.15 `virtual void Ipopt::CompoundVector::SetImpl (Number value)` [protected],[virtual]

Set each element in the vector to the scalar alpha.

Implements [Ipopt::Vector](#).

6.22.3.16 `virtual void Ipopt::CompoundVector::ElementWiseDivideImpl (const Vector & x)` [protected],[virtual]

Element-wise division $y_i \leftarrow y_i / x_i$.

Implements [Ipopt::Vector](#).

6.22.3.17 `virtual void Ipopt::CompoundVector::ElementWiseMultiplyImpl (const Vector & x)` [protected],[virtual]

Element-wise multiplication $y_i \leftarrow y_i * x_i$.

Implements [Ipopt::Vector](#).

6.22.3.18 `virtual void Ipopt::CompoundVector::ElementWiseMaxImpl (const Vector & x)` [protected],[virtual]

Element-wise max against entries in x.

Implements [Ipopt::Vector](#).

6.22.3.19 `virtual void lpopt::CompoundVector::ElementWiseMinImpl (const Vector & x) [protected],[virtual]`

Element-wise min against entries in x.

Implements [lpopt::Vector](#).

6.22.3.20 `virtual void lpopt::CompoundVector::ElementWiseReciprocalImpl () [protected],[virtual]`

Element-wise reciprocal.

Implements [lpopt::Vector](#).

6.22.3.21 `virtual void lpopt::CompoundVector::ElementWiseAbsImpl () [protected],[virtual]`

Element-wise absolute values.

Implements [lpopt::Vector](#).

6.22.3.22 `virtual void lpopt::CompoundVector::ElementWiseSqrtImpl () [protected],[virtual]`

Element-wise square-root.

Implements [lpopt::Vector](#).

6.22.3.23 `virtual void lpopt::CompoundVector::ElementWiseSgnImpl () [protected],[virtual]`

Replaces entries with sgn of the entry.

Implements [lpopt::Vector](#).

6.22.3.24 `virtual void lpopt::CompoundVector::AddScalarImpl (Number scalar) [protected],[virtual]`

Add scalar to every component of the vector.

Implements [lpopt::Vector](#).

6.22.3.25 `virtual Number lpopt::CompoundVector::MaxImpl () const [protected],[virtual]`

Max value in the vector.

Implements [lpopt::Vector](#).

6.22.3.26 `virtual Number lpopt::CompoundVector::MinImpl () const [protected],[virtual]`

Min value in the vector.

Implements [lpopt::Vector](#).

6.22.3.27 `virtual Number lpopt::CompoundVector::SumImpl () const [protected],[virtual]`

Computes the sum of the elements of vector.

Implements [lpopt::Vector](#).

6.22.3.28 `virtual Number lpopt::CompoundVector::SumLogsImpl () const [protected],[virtual]`

Computes the sum of the logs of the elements of vector.

Implements [lpopt::Vector](#).

6.22.3.29 `void Ipopt::CompoundVector::AddTwoVectorsImpl (Number a, const Vector & v1, Number b, const Vector & v2, Number c)` [protected],[virtual]

Add two vectors ($a * v1 + b * v2$).

Result is stored in this vector.

Reimplemented from [Ipopt::Vector](#).

6.22.3.30 `Number Ipopt::CompoundVector::FracToBoundImpl (const Vector & delta, Number tau) const` [protected],[virtual]

Fraction to the boundary parameter.

Reimplemented from [Ipopt::Vector](#).

6.22.3.31 `void Ipopt::CompoundVector::AddVectorQuotientImpl (Number a, const Vector & z, const Vector & s, Number c)` [protected],[virtual]

Add the quotient of two vectors, $y = a * z/s + c * y$.

Reimplemented from [Ipopt::Vector](#).

6.22.3.32 `virtual bool Ipopt::CompoundVector::IsValidNumbersImpl () const` [protected],[virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Vector](#).

6.22.3.33 `virtual void Ipopt::CompoundVector::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected],[virtual]

Print the entire vector.

Implements [Ipopt::Vector](#).

6.22.3.34 `void Ipopt::CompoundVector::operator= (const CompoundVector &)` [private]

Overloaded Equals Operator.

6.22.3.35 `bool Ipopt::CompoundVector::VectorsValid ()` [private]

6.22.3.36 `const Vector * Ipopt::CompoundVector::ConstComp (Index i) const` [inline],[private]

Definition at line 314 of file `IpCompoundVector.hpp`.

6.22.3.37 `Vector * Ipopt::CompoundVector::Comp (Index i)` [inline],[private]

Definition at line 330 of file `IpCompoundVector.hpp`.

6.22.4 Member Data Documentation

6.22.4.1 `std::vector< SmartPtr<Vector> > Ipopt::CompoundVector::comps_` [private]

Components of the compound vector.

The components are stored by SmartPtrs in a `std::vector`

Definition at line 219 of file `IpCompoundVector.hpp`.

6.22.4.2 `std::vector< SmartPtr<const Vector> > Ipopt::CompoundVector::const_comps_` [private]

Definition at line 220 of file `IpCompoundVector.hpp`.

6.22.4.3 `const CompoundVectorSpace* Ipopt::CompoundVector::owner_space_` [private]

Definition at line 222 of file `IpCompoundVector.hpp`.

6.22.4.4 `bool Ipopt::CompoundVector::vectors_valid_` [private]

Definition at line 224 of file `IpCompoundVector.hpp`.

The documentation for this class was generated from the following file:

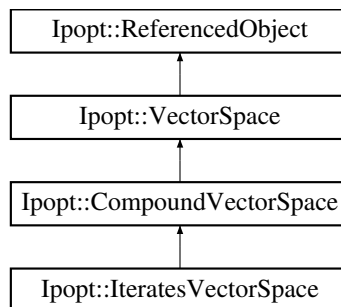
- [LinAlg/IpCompoundVector.hpp](#)

6.23 Ipopt::CompoundVectorSpace Class Reference

This vectors space is the vector space for [CompoundVector](#).

```
#include <IpCompoundVector.hpp>
```

Inheritance diagram for `Ipopt::CompoundVectorSpace`:



Public Member Functions

- virtual void [SetCompSpace](#) ([Index](#) icomp, const [VectorSpace](#) &vec_space)
Method for setting the individual component VectorSpaces.
- [SmartPtr](#)< const [VectorSpace](#) > [GetCompSpace](#) ([Index](#) icomp) const
Method for obtaining an individual component VectorSpace.
- [Index](#) [NCompSpaces](#) () const
Accessor method to obtain the number of components.
- virtual [CompoundVector](#) * [MakeNewCompoundVector](#) (bool create_new=true) const
Method for creating a new vector of this specific type.
- virtual [Vector](#) * [MakeNew](#) () const
Overloaded MakeNew method for the VectorSpace base class.

Constructors/Destructors.

- [CompoundVectorSpace](#) ([Index](#) ncomp_spaces, [Index](#) total_dim)
Constructor, has to be given the number of components and the total dimension of all components combined.
- [~CompoundVectorSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [CompoundVectorSpace](#) ()
Default constructor.
- [CompoundVectorSpace](#) (const [CompoundVectorSpace](#) &)
Copy Constructor.
- [CompoundVectorSpace](#) & operator= (const [CompoundVectorSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- const [Index](#) ncomp_spaces_
Number of components.
- std::vector< [SmartPtr](#)< const [VectorSpace](#) > > comp_spaces_
std::vector of vector spaces for the components

6.23.1 Detailed Description

This vectors space is the vector space for [CompoundVector](#).

Before a [CompoundVector](#) can be created, all components of this [CompoundVectorSpace](#) have to be set. When calling the constructor, the number of component has to be specified. The individual VectorSpaces can be set with the SetComp method.

Definition at line 239 of file IpCompoundVector.hpp.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Ipopt::CompoundVectorSpace::CompoundVectorSpace ([Index](#) ncomp_spaces, [Index](#) total_dim)

Constructor, has to be given the number of components and the total dimension of all components combined.

6.23.2.2 Ipopt::CompoundVectorSpace::~~CompoundVectorSpace () [inline]

Destructor.

Definition at line 249 of file IpCompoundVector.hpp.

6.23.2.3 Ipopt::CompoundVectorSpace::CompoundVectorSpace () [private]

Default constructor.

6.23.2.4 Ipopt::CompoundVectorSpace::CompoundVectorSpace (const [CompoundVectorSpace](#) &) [private]

Copy Constructor.

6.23.3 Member Function Documentation

6.23.3.1 `virtual void Ipopt::CompoundVectorSpace::SetCompSpace (Index icomp, const VectorSpace & vec_space)`
[virtual]

Method for setting the individual component VectorSpaces.

Parameters

<i>icomp</i>	Number of the component to be set
<i>vec_space</i>	VectorSpace for component icomp

Reimplemented in [Ipopt::IteratesVectorSpace](#).

6.23.3.2 `SmartPointer<const VectorSpace> Ipopt::CompoundVectorSpace::GetCompSpace (Index icomp) const`

Method for obtaining an individual component [VectorSpace](#).

6.23.3.3 `Index Ipopt::CompoundVectorSpace::NCompSpaces () const` `[inline]`

Accessor method to obtain the number of components.

Definition at line 262 of file `IpCompoundVector.hpp`.

6.23.3.4 `virtual CompoundVector* Ipopt::CompoundVectorSpace::MakeNewCompoundVector (bool create_new = true) const` `[inline]`, `[virtual]`

Method for creating a new vector of this specific type.

Reimplemented in [Ipopt::IteratesVectorSpace](#).

Definition at line 268 of file `IpCompoundVector.hpp`.

6.23.3.5 `virtual Vector* Ipopt::CompoundVectorSpace::MakeNew () const` `[inline]`, `[virtual]`

Overloaded MakeNew method for the [VectorSpace](#) base class.

Implements [Ipopt::VectorSpace](#).

Reimplemented in [Ipopt::IteratesVectorSpace](#).

Definition at line 275 of file `IpCompoundVector.hpp`.

6.23.3.6 `CompoundVectorSpace& Ipopt::CompoundVectorSpace::operator= (const CompoundVectorSpace &)` `[private]`

Overloaded Equals Operator.

6.23.4 Member Data Documentation

6.23.4.1 `const Index Ipopt::CompoundVectorSpace::ncomp_spaces_` `[private]`

Number of components.

Definition at line 300 of file `IpCompoundVector.hpp`.

6.23.4.2 `std::vector< SmartPtr<const VectorSpace> > Ipopt::CompoundVectorSpace::comp_spaces_` `[private]`

`std::vector` of vector spaces for the components

Definition at line 303 of file `IpCompoundVector.hpp`.

The documentation for this class was generated from the following file:

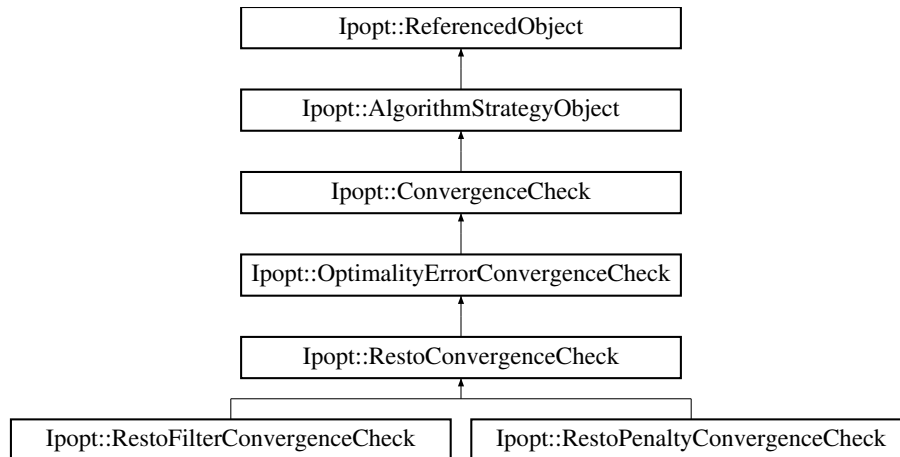
- [LinAlg/IpCompoundVector.hpp](#)

6.24 Ipopt::ConvergenceCheck Class Reference

Base class for checking the algorithm termination criteria.

```
#include <IpConvCheck.hpp>
```

Inheritance diagram for Ipopt::ConvergenceCheck:



Public Types

- enum [ConvergenceStatus](#) {
[CONTINUE](#), [CONVERGED](#), [CONVERGED_TO_ACCEPTABLE_POINT](#), [MAXITER_EXCEEDED](#),
[CPUTIME_EXCEEDED](#), [DIVERGING](#), [USER_STOP](#), [FAILED](#) }
Convergence return enum.

Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)
- virtual [ConvergenceStatus](#) [CheckConvergence](#) (bool call_intermediate_callback=true)=0
Pure virtual method for performing the convergence test.
- virtual bool [CurrentIsAcceptable](#) ()=0
Method for testing if the current iterate is considered to satisfy the "acceptable level" of accuracy.

Constructors/Destructors

- [ConvergenceCheck](#) ()
Constructor.
- virtual [~ConvergenceCheck](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ConvergenceCheck](#) (const [ConvergenceCheck](#) &)
Default Constructor.
- void `operator=` (const [ConvergenceCheck](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.24.1 Detailed Description

Base class for checking the algorithm termination criteria.

Definition at line 20 of file IpConvCheck.hpp.

6.24.2 Member Enumeration Documentation

6.24.2.1 enum Ipopt::ConvergenceCheck::ConvergenceStatus

Convergence return enum.

Enumerator

CONTINUE
CONVERGED
CONVERGED_TO_ACCEPTABLE_POINT
MAXITER_EXCEEDED
CPUTIME_EXCEEDED
DIVERGING
USER_STOP
FAILED

Definition at line 35 of file IpConvCheck.hpp.

6.24.3 Constructor & Destructor Documentation

6.24.3.1 Ipopt::ConvergenceCheck::ConvergenceCheck () [inline]

Constructor.

Definition at line 26 of file IpConvCheck.hpp.

6.24.3.2 virtual Ipopt::ConvergenceCheck::~~ConvergenceCheck () [inline],[virtual]

Default destructor.

Definition at line 30 of file IpConvCheck.hpp.

6.24.3.3 Ipopt::ConvergenceCheck::ConvergenceCheck (const [ConvergenceCheck](#) &) [private]

Default Constructor.

Copy Constructor

6.24.4 Member Function Documentation

6.24.4.1 `virtual bool Ipopt::ConvergenceCheck::InitializeImpl (const OptionsList & options, const std::string & prefix)` [pure virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::RestoFilterConvergenceCheck](#), [Ipopt::RestoPenaltyConvergenceCheck](#), [Ipopt::RestoConvergenceCheck](#), and [Ipopt::OptimalityErrorConvergenceCheck](#).

6.24.4.2 `virtual ConvergenceStatus Ipopt::ConvergenceCheck::CheckConvergence (bool call_intermediate_callback = true)` [pure virtual]

Pure virtual method for performing the convergence test.

If `call_intermediate_callback` is true, the user callback method in the [NLP](#) should be called in order to see if the user requests an early termination.

Implemented in [Ipopt::RestoConvergenceCheck](#), and [Ipopt::OptimalityErrorConvergenceCheck](#).

6.24.4.3 `virtual bool Ipopt::ConvergenceCheck::CurrentIsAcceptable ()` [pure virtual]

Method for testing if the current iterate is considered to satisfy the "acceptable level" of accuracy.

The idea is that if the desired convergence tolerance cannot be achieved, the algorithm might stop after a number of acceptable points have been encountered.

Implemented in [Ipopt::OptimalityErrorConvergenceCheck](#).

6.24.4.4 `void Ipopt::ConvergenceCheck::operator= (const ConvergenceCheck &)` [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

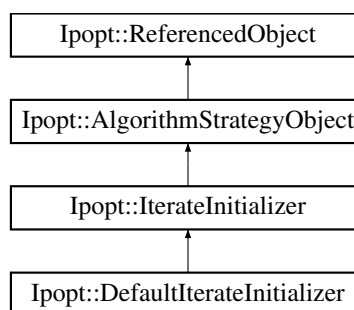
- [Algorithm/lpConvCheck.hpp](#)

6.25 Ipopt::DefaultIterateInitializer Class Reference

Class implementing the default initialization procedure (based on user options) for the iterates.

`#include <IpDefaultIterateInitializer.hpp>`

Inheritance diagram for [Ipopt::DefaultIterateInitializer](#):



Public Types

Enums of option values

- enum [BoundMultInitMethod](#) { [B_CONSTANT](#) =0, [B_MU_BASED](#) }

Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [SetInitialIterates](#) ()
Compute the initial iterates and set the into the curr field of the ip_data object.

Constructors/Destructors

- [DefaultIterateInitializer](#) (const [SmartPtr](#)< [EqMultiplierCalculator](#) > &eq_mult_calculator, const [SmartPtr](#)< [IterateInitializer](#) > &warm_start_initializer, const [SmartPtr](#)< [AugSystemSolver](#) > aug_system_solver=NULL)
Constructor.
- virtual [~DefaultIterateInitializer](#) ()
Default destructor.

Static Public Member Functions

- static void [push_variables](#) (const [Journalist](#) &jnlst, [Number](#) bound_push, [Number](#) bound_frac, std::string name, const [Vector](#) &orig_x, [SmartPtr](#)< const [Vector](#) > &new_x, const [Vector](#) &x_L, const [Vector](#) &x_U, const [Matrix](#) &Px_L, const [Matrix](#) &Px_U)
Auxilliary function for moving the initial point.
- static void [least_square_mults](#) (const [Journalist](#) &jnlst, [IpoptNLP](#) &ip_nlp, [IpoptData](#) &ip_data, [IpoptCalculatedQuantities](#) &ip_cq, const [SmartPtr](#)< [EqMultiplierCalculator](#) > &eq_mult_calculator, [Number](#) constr_mult_init_max)
Auxilliary function for computing least_square multipliers.
- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > reg_options)
Methods for IpoptType.

Private Member Functions

- bool [CalculateLeastSquarePrimals](#) ([Vector](#) &x_ls, [Vector](#) &s_ls)
Auxilliary method for computing least square primal variables.
- bool [CalculateLeastSquareDuals](#) ([Vector](#) &zL_new, [Vector](#) &zU_new, [Vector](#) &vL_new, [Vector](#) &vU_new, [Vector](#) &yc_new, [Vector](#) &yd_new)
Auxilliary method for computing least square dual variables.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [DefaultIterateInitializer](#) ()
Default Constructor.

- [DefaultIterateInitializer](#) (const [DefaultIterateInitializer](#) &)
Copy Constructor.
- void [operator=](#) (const [DefaultIterateInitializer](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [EqMultiplierCalculator](#) > [eq_mult_calculator_](#)
object to be used for the initialization of the equality constraint multipliers.
- [SmartPtr](#)< [IterateInitializer](#) > [warm_start_initializer_](#)
object to be used for a warm start initialization
- [SmartPtr](#)< [AugSystemSolver](#) > [aug_system_solver_](#)
Object for solving the augmented system.

Algorithmic Parameters

- [Number](#) [bound_push_](#)
Absolute parameter for bumping x0.
- [Number](#) [bound_frac_](#)
Relative parameter for bumping x0.
- [Number](#) [slack_bound_push_](#)
Absolute parameter for bumping s0.
- [Number](#) [slack_bound_frac_](#)
Relative parameter for bumping s0.
- [Number](#) [constr_mult_init_max_](#)
If max-norm of the initial equality constraint multiplier estimate is larger than this, the initial y_ variables are set to zero.*
- [Number](#) [bound_mult_init_val_](#)
Initial value for all bound multipliers.
- bool [warm_start_init_point_](#)
Flag indicating whether warm_start_initializer should be used instead of the default initialization.
- bool [least_square_init_primal_](#)
Flag indicating whether the primal variables should be initialized as least square fit for the linearized constraints.
- bool [least_square_init_duals_](#)
Flag indicating whether all dual variables should be initialized as least square fit for the linearized dual infeasibility.
- [BoundMultInitMethod](#) [bound_mult_init_method_](#)
Flag indicating how bound multipliers are initialized.
- [Number](#) [mu_init_](#)
Initial value of barrier parameter.

Additional Inherited Members

6.25.1 Detailed Description

Class implementing the default initialization procedure (based on user options) for the iterates.

It is used at the very beginning of the optimization for determine the starting point for all variables.

Definition at line 24 of file `IpDefaultIterateInitializer.hpp`.

6.25.2 Member Enumeration Documentation

6.25.2.1 enum Ipopt::DefaultIterateInitializer::BoundMultInitMethod

Enumerator

B_CONSTANT

B_MU_BASED

Definition at line 88 of file IpDefaultIterateInitializer.hpp.

6.25.3 Constructor & Destructor Documentation

6.25.3.1 `Ipopt::DefaultIterateInitializer::DefaultIterateInitializer (const SmartPtr< EqMultiplierCalculator > & eq_mult_calculator, const SmartPtr< IterateInitializer > & warm_start_initializer, const SmartPtr< AugSystemSolver > aug_system_solver = NULL)`

Constructor.

If eq_mult_calculator is not NULL, it will be used to compute the initial values for equality constraint multipliers. If warm_start_initializer is not NULL, it will be used to compute the initial values if the option warm_start_init_point is chosen.

6.25.3.2 `virtual Ipopt::DefaultIterateInitializer::~~DefaultIterateInitializer () [inline], [virtual]`

Default destructor.

Definition at line 40 of file IpDefaultIterateInitializer.hpp.

6.25.3.3 `Ipopt::DefaultIterateInitializer::DefaultIterateInitializer () [private]`

Default Constructor.

6.25.3.4 `Ipopt::DefaultIterateInitializer::DefaultIterateInitializer (const DefaultIterateInitializer &) [private]`

Copy Constructor.

6.25.4 Member Function Documentation

6.25.4.1 `virtual bool Ipopt::DefaultIterateInitializer::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::IterateInitializer](#).

6.25.4.2 `virtual bool Ipopt::DefaultIterateInitializer::SetInitialIterates () [virtual]`

Compute the initial iterates and set the into the curr field of the ip_data object.

Implements [Ipopt::IterateInitializer](#).

6.25.4.3 `static void Ipopt::DefaultIterateInitializer::push_variables (const Journalist & jnlst, Number bound_push, Number bound_frac, std::string name, const Vector & orig_x, SmartPtr< const Vector > & new_x, const Vector & x_L, const Vector & x_U, const Matrix & Px_L, const Matrix & Px_U) [static]`

Auxilliary function for moving the initial point.

This is declared static so that it can also be used from [WarmStartIterateInitializer](#).

```
6.25.4.4 static void Ipopt::DefaultIterateInitializer::least_square_mults ( const Journalist & jnlst, IpoptNLP & ip_nlp,
    IpoptData & ip_data, IpoptCalculatedQuantities & ip_cq, const SmartPtr< EqMultiplierCalculator > &
    eq_mult_calculator, Number constr_mult_init_max ) [static]
```

Auxilliary function for computing least_square multipliers.

The multipliers are computed based on the values in the trial fields (current is overwritten). On return, the multipliers are in the trial fields as well. The value of constr_mult_init_max determines if the computed least square estimate should be used, or if the initial multipliers are set to zero.

```
6.25.4.5 static void Ipopt::DefaultIterateInitializer::RegisterOptions ( SmartPtr< RegisteredOptions > reg_options )
    [static]
```

Methods for IpoptType.

```
6.25.4.6 void Ipopt::DefaultIterateInitializer::operator= ( const DefaultIterateInitializer & ) [private]
```

Overloaded Equals Operator.

```
6.25.4.7 bool Ipopt::DefaultIterateInitializer::CalculateLeastSquarePrimals ( Vector & x_ls, Vector & s_ls ) [private]
```

Auxilliary method for computing least square primal variables.

```
6.25.4.8 bool Ipopt::DefaultIterateInitializer::CalculateLeastSquareDuals ( Vector & zL_new, Vector & zU_new, Vector & vL_new,
    Vector & vU_new, Vector & yc_new, Vector & yd_new ) [private]
```

Auxilliary method for computing least square dual variables.

6.25.5 Member Data Documentation

```
6.25.5.1 Number Ipopt::DefaultIterateInitializer::bound_push_ [private]
```

Absolute parameter for bumping x0.

Definition at line 116 of file IpDefaultIterateInitializer.hpp.

```
6.25.5.2 Number Ipopt::DefaultIterateInitializer::bound_frac_ [private]
```

Relative parameter for bumping x0.

Definition at line 118 of file IpDefaultIterateInitializer.hpp.

```
6.25.5.3 Number Ipopt::DefaultIterateInitializer::slack_bound_push_ [private]
```

Absolute parameter for bumping s0.

Definition at line 120 of file IpDefaultIterateInitializer.hpp.

```
6.25.5.4 Number Ipopt::DefaultIterateInitializer::slack_bound_frac_ [private]
```

Relative parameter for bumping s0.

Definition at line 122 of file IpDefaultIterateInitializer.hpp.

6.25.5.5 `Number Ipopt::DefaultIterateInitializer::constr_mult_init_max_ [private]`

If max-norm of the initial equality constraint multiplier estimate is larger than this, the initial y_* variables are set to zero.
Definition at line 127 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.6 `Number Ipopt::DefaultIterateInitializer::bound_mult_init_val_ [private]`

Initial value for all bound multipliers.

Definition at line 129 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.7 `bool Ipopt::DefaultIterateInitializer::warm_start_init_point_ [private]`

Flag indicating whether `warm_start_initializer` should be used instead of the default initialization.

Definition at line 132 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.8 `bool Ipopt::DefaultIterateInitializer::least_square_init_primal_ [private]`

Flag indicating whether the primal variables should be initialized as least square fit for the linearized constraints.

Definition at line 136 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.9 `bool Ipopt::DefaultIterateInitializer::least_square_init_duals_ [private]`

Flag indicating whether all dual variables should be initialized as least square fit for the linearized dual infeasibility.

Definition at line 140 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.10 `BoundMultInitMethod Ipopt::DefaultIterateInitializer::bound_mult_init_method_ [private]`

Flag indicating how bound multipliers are initialized.

Definition at line 142 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.11 `Number Ipopt::DefaultIterateInitializer::mu_init_ [private]`

Initial value of barrier parameter.

Definition at line 144 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.12 `SmartPtr<EqMultiplierCalculator> Ipopt::DefaultIterateInitializer::eq_mult_calculator_ [private]`

object to be used for the initialization of the equality constraint multipliers.

Definition at line 149 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.13 `SmartPtr<IterateInitializer> Ipopt::DefaultIterateInitializer::warm_start_initializer_ [private]`

object to be used for a warm start initialization

Definition at line 152 of file `IpDefaultIterateInitializer.hpp`.

6.25.5.14 `SmartPtr<AugSystemSolver> Ipopt::DefaultIterateInitializer::aug_system_solver_ [private]`

Object for solving the augmented system.

This is only required if we use the least square initialization of primal and all dual variables.

Definition at line 157 of file `IpDefaultIterateInitializer.hpp`.

The documentation for this class was generated from the following file:

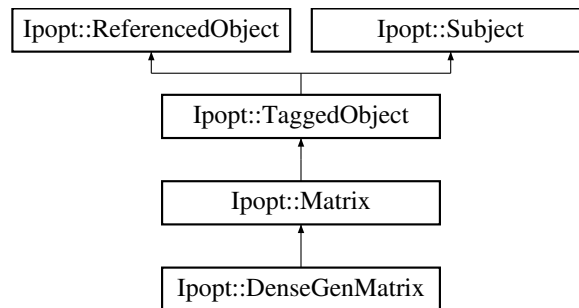
- [Algorithm/lpDefaultIterateInitializer.hpp](#)

6.26 Ipopt::DenseGenMatrix Class Reference

Class for dense general matrices.

```
#include <IpDenseGenMatrix.hpp>
```

Inheritance diagram for Ipopt::DenseGenMatrix:



Public Member Functions

- [SmartPointer< DenseGenMatrix > MakeNewDenseGenMatrix \(\)](#) const
Create a new [DenseGenMatrix](#) from same [MatrixSpace](#).
- [Number * Values \(\)](#)
Retrieve the array for storing the matrix elements.
- [const Number * Values \(\)](#) const
Retrieve the array that stores the matrix elements.
- [void Copy \(const DenseGenMatrix &M\)](#)
Method for copying the content of another matrix into this matrix.
- [void FillIdentity \(Number factor=1.\)](#)
Set this matrix to be a multiple of the identity matrix .
- [void ScaleColumns \(const DenseVector &scal_vec\)](#)
Method for scaling the columns of the matrix.
- [void AddMatrixProduct \(Number alpha, const DenseGenMatrix &A, bool transA, const DenseGenMatrix &B, bool transB, Number beta\)](#)
Method for adding the product of two matrices to this matrix.
- [void HighRankUpdateTranspose \(Number alpha, const MultiVectorMatrix &V1, const MultiVectorMatrix &V2, Number beta\)](#)
Method for adding a high-rank update to this matrix.
- [bool ComputeCholeskyFactor \(const DenseSymMatrix &M\)](#)
Method for computing the Cholesky factorization of a positive definite matrix.
- [bool ComputeEigenVectors \(const DenseSymMatrix &M, DenseVector &Evalues\)](#)
Method for computing an eigenvalue decomposition of the given symmetrix matrix M.
- [void CholeskyBackSolveMatrix \(bool trans, Number alpha, DenseGenMatrix &B\)](#) const
Method for performing one backsolve with an entire matrix on the right hand side, assuming that the this matrix is square and contains a lower triangular matrix.
- [void CholeskySolveVector \(DenseVector &b\)](#) const

Method for performing a solve of a linear system for one vector, assuming that this matrix contains the Cholesky factor for the linear system.

- void [CholeskySolveMatrix](#) ([DenseGenMatrix](#) &B) const

Method for performing a solve of a linear system for one right-hand-side matrix, assuming that this matrix contains the Cholesky factor for the linear system.

- bool [ComputeLUFactorInPlace](#) ()

Method for computing the LU factorization of an unsymmetric matrix.

- void [LUSolveMatrix](#) ([DenseGenMatrix](#) &B) const

Method for using a previously computed LU factorization for a backsolve with a matrix on the rhs.

- void [LUSolveVector](#) ([DenseVector](#) &b) const

Method for using a previously computed LU factorization for a backsolve with a single vector.

Constructors / Destructors

- [DenseGenMatrix](#) (const [DenseGenMatrixSpace](#) *owner_space)

Constructor, taking the owner_space.

- [~DenseGenMatrix](#) ()

Destructor.

Protected Member Functions

Overloaded methods from Matrix base class

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual void [TransMultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix(transpose) vector multiply.
- virtual bool [IsValidNumbersImpl](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &cols_norms, bool init) const
Compute the max-norm of the columns in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Types

- enum [Factorization](#) { [NONE](#), [LU](#), [CHOL](#) }
Enum for factorization type.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [DenseGenMatrix](#) ()
Default Constructor.

- [DenseGenMatrix](#) (const [DenseGenMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [DenseGenMatrix](#) &)
Overloaded Equals Operator.

Private Attributes

- const [DenseGenMatrixSpace](#) * [owner_space_](#)
- [Number](#) * [values_](#)
Array for storing the matrix elements (one columns after each other)
- bool [initialized_](#)
Flag indicating whether the [values_](#) array has been initialized.
- [Factorization](#) [factorization_](#)
Flag indicating if and which factorization has been applied.
- int * [pivot_](#)
Array for storing the pivot sequences if the matrix has been LU-factorized.

Additional Inherited Members

6.26.1 Detailed Description

Class for dense general matrices.

[Matrix](#) elements are stored in one array in "Fortran" format.

Definition at line 26 of file `IpDenseGenMatrix.hpp`.

6.26.2 Member Enumeration Documentation

6.26.2.1 `enum Ipopt::DenseGenMatrix::Factorization` `[private]`

Enum for factorization type.

Enumerator

NONE
LU
CHOL

Definition at line 192 of file `IpDenseGenMatrix.hpp`.

6.26.3 Constructor & Destructor Documentation

6.26.3.1 `Ipopt::DenseGenMatrix::DenseGenMatrix (const DenseGenMatrixSpace * owner_space)`

Constructor, taking the `owner_space`.

6.26.3.2 `Ipopt::DenseGenMatrix::~~DenseGenMatrix ()`

Destructor.

6.26.3.3 Ipopt::DenseGenMatrix::DenseGenMatrix () [private]

Default Constructor.

6.26.3.4 Ipopt::DenseGenMatrix::DenseGenMatrix (const DenseGenMatrix &) [private]

Copy Constructor.

6.26.4 Member Function Documentation

6.26.4.1 SmartPtr< DenseGenMatrix > Ipopt::DenseGenMatrix::MakeNewDenseGenMatrix () const [inline]

Create a new [DenseGenMatrix](#) from same [MatrixSpace](#).

Definition at line 239 of file IpDenseGenMatrix.hpp.

6.26.4.2 Number* Ipopt::DenseGenMatrix::Values () [inline]

Retrieve the array for storing the matrix elements.

This is the non-const version, and it is assume that afterwards the calling method will set all matrix elements. The matrix elements are stored one column after each other.

Definition at line 48 of file IpDenseGenMatrix.hpp.

6.26.4.3 const Number* Ipopt::DenseGenMatrix::Values () const [inline]

Retrieve the array that stores the matrix elements.

This is the const version, i.e., read-only. The matrix elements are stored one column after each other.

Definition at line 58 of file IpDenseGenMatrix.hpp.

6.26.4.4 void Ipopt::DenseGenMatrix::Copy (const DenseGenMatrix & M)

Method for copying the content of another matrix into this matrix.

6.26.4.5 void Ipopt::DenseGenMatrix::FillIdentity (Number factor = 1 .)

Set this matrix to be a multiple of the identity matrix .

This assumes that this matrix is square.

6.26.4.6 void Ipopt::DenseGenMatrix::ScaleColumns (const DenseVector & scal_vec)

Method for scaling the columns of the matrix.

The scaling factors are given in form of a [DenseVector](#)

6.26.4.7 void Ipopt::DenseGenMatrix::AddMatrixProduct (Number alpha, const DenseGenMatrix & A, bool transA, const DenseGenMatrix & B, bool transB, Number beta)

Method for adding the product of two matrices to this matrix.

6.26.4.8 void Ipopt::DenseGenMatrix::HighRankUpdateTranspose (Number alpha, const MultiVectorMatrix & V1, const MultiVectorMatrix & V2, Number beta)

Method for adding a high-rank update to this matrix.

It computes $M = \alpha * V1^T V2 + \beta * M$, where V1 and V2 are MultiVectorMatrices.

6.26.4.9 `bool Ipopt::DenseGenMatrix::ComputeCholeskyFactor (const DenseSymMatrix & M)`

Method for computing the Cholesky factorization of a positive definite matrix.

The factor is stored in this matrix, as lower-triangular matrix, i.e., $M = J * J^T$. The return value is false if the factorization could not be done, e.g., when the matrix is not positive definite.

6.26.4.10 `bool Ipopt::DenseGenMatrix::ComputeEigenVectors (const DenseSymMatrix & M, DenseVector & Eigenvalues)`

Method for computing an eigenvalue decomposition of the given symmetric matrix M.

On return, this matrix contains the eigenvalues in its columns, and Eigenvalues contains the eigenvalues. The return value is false, if there are problems during the computation.

6.26.4.11 `void Ipopt::DenseGenMatrix::CholeskyBackSolveMatrix (bool trans, Number alpha, DenseGenMatrix & B) const`

Method for performing one backsolve with an entire matrix on the right hand side, assuming that this matrix is square and contains a lower triangular matrix.

The incoming right hand side B is overwritten with the solution X of $op(A)*X = alpha*B$. $op(A) = A$ or $op(A) = A^T$.

6.26.4.12 `void Ipopt::DenseGenMatrix::CholeskySolveVector (DenseVector & b) const`

Method for performing a solve of a linear system for one vector, assuming that this matrix contains the Cholesky factor for the linear system.

The vector b contains the right hand side on input, and contains the solution on output.

6.26.4.13 `void Ipopt::DenseGenMatrix::CholeskySolveMatrix (DenseGenMatrix & B) const`

Method for performing a solve of a linear system for one right-hand-side matrix, assuming that this matrix contains the Cholesky factor for the linear system.

The matrix B contains the right hand sides on input, and contains the solution on output.

6.26.4.14 `bool Ipopt::DenseGenMatrix::ComputeLUFactorInPlace ()`

Method for computing the LU factorization of an unsymmetric matrix.

The factorization is done in place.

6.26.4.15 `void Ipopt::DenseGenMatrix::LUSolveMatrix (DenseGenMatrix & B) const`

Method for using a previously computed LU factorization for a backsolve with a matrix on the rhs.

6.26.4.16 `void Ipopt::DenseGenMatrix::LUSolveVector (DenseVector & b) const`

Method for using a previously computed LU factorization for a backsolve with a single vector.

6.26.4.17 `virtual void Ipopt::DenseGenMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix-vector multiply.

Computes $y = alpha * Matrix * x + beta * y$

Implements `Ipopt::Matrix`.

6.26.4.18 `virtual void Ipopt::DenseGenMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const` [protected], [virtual]

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.26.4.19 `virtual bool Ipopt::DenseGenMatrix::IsValidNumbersImpl () const` [protected], [virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.26.4.20 `virtual void Ipopt::DenseGenMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` [protected], [virtual]

Compute the max-norm of the rows in the matrix.

The result is stored in *rows_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.26.4.21 `virtual void Ipopt::DenseGenMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const` [protected], [virtual]

Compute the max-norm of the columns in the matrix.

The result is stored in *cols_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.26.4.22 `virtual void Ipopt::DenseGenMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected], [virtual]

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.26.4.23 `void Ipopt::DenseGenMatrix::operator= (const DenseGenMatrix &)` [private]

Overloaded Equals Operator.

6.26.5 Member Data Documentation

6.26.5.1 `const DenseGenMatrixSpace* Ipopt::DenseGenMatrix::owner_space_` [private]

Definition at line 182 of file `IpDenseGenMatrix.hpp`.

6.26.5.2 `Number* Ipopt::DenseGenMatrix::values_` [private]

Array for storing the matrix elements (one columns after each other)

Definition at line 186 of file `IpDenseGenMatrix.hpp`.

6.26.5.3 `bool Ipopt::DenseGenMatrix::initialized_` [private]

Flag indicating whether the *values_* array has been initialized.

Definition at line 189 of file `IpDenseGenMatrix.hpp`.

6.26.5.4 Factorization `Ipopt::DenseGenMatrix::factorization_` [private]

Flag indicating if and which factorization has been applied.

Definition at line 200 of file `IpDenseGenMatrix.hpp`.

6.26.5.5 `int* Ipopt::DenseGenMatrix::pivot_` [private]

Array for storing the pivot sequences if the matrix has been LU-factorized.

Definition at line 203 of file `IpDenseGenMatrix.hpp`.

The documentation for this class was generated from the following file:

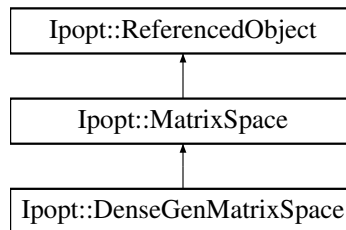
- [LinAlg/IpDenseGenMatrix.hpp](#)

6.27 `Ipopt::DenseGenMatrixSpace` Class Reference

This is the matrix space for [DenseGenMatrix](#).

```
#include <IpDenseGenMatrix.hpp>
```

Inheritance diagram for `Ipopt::DenseGenMatrixSpace`:



Public Member Functions

- [DenseGenMatrix](#) * [MakeNewDenseGenMatrix](#) () const
Method for creating a new matrix of this specific type.
- virtual [Matrix](#) * [MakeNew](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.

Constructors / Destructors

- [DenseGenMatrixSpace](#) ([Index](#) nRows, [Index](#) nCols)
Constructor for matrix space for DenseGenMatrices.
- [~DenseGenMatrixSpace](#) ()
Destructor.

6.27.1 Detailed Description

This is the matrix space for [DenseGenMatrix](#).

Definition at line 208 of file `IpDenseGenMatrix.hpp`.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Ipopt::DenseGenMatrixSpace::DenseGenMatrixSpace (Index *nRows*, Index *nCols*)

Constructor for matrix space for DenseGenMatrices.

Takes in dimension of the matrices.

6.27.2.2 Ipopt::DenseGenMatrixSpace::~~DenseGenMatrixSpace () [inline]

Destructor.

Definition at line 219 of file IpDenseGenMatrix.hpp.

6.27.3 Member Function Documentation

6.27.3.1 DenseGenMatrix* Ipopt::DenseGenMatrixSpace::MakeNewDenseGenMatrix () const [inline]

Method for creating a new matrix of this specific type.

Definition at line 224 of file IpDenseGenMatrix.hpp.

6.27.3.2 virtual Matrix* Ipopt::DenseGenMatrixSpace::MakeNew () const [inline],[virtual]

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 231 of file IpDenseGenMatrix.hpp.

The documentation for this class was generated from the following file:

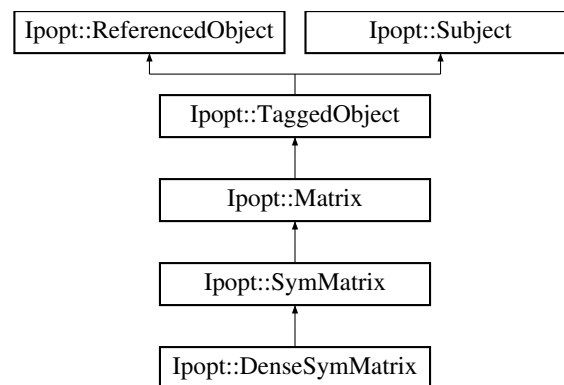
- [LinAlg/IpDenseGenMatrix.hpp](#)

6.28 Ipopt::DenseSymMatrix Class Reference

Class for dense symetrix matrices.

```
#include <IpDenseSymMatrix.hpp>
```

Inheritance diagram for Ipopt::DenseSymMatrix:



Public Member Functions

- `SmartPtr< DenseSymMatrix > MakeNewDenseSymMatrix () const`
Create a new `DenseSymMatrix` from same `MatrixSpace`.
- `Number * Values ()`
Retrieve the array for storing the matrix elements.
- `const Number * Values () const`
Retrieve the array that stores the matrix elements.
- `void FillIdentity (Number factor=1.)`
Set this matrix to be a multiple of the identity matrix.
- `void AddMatrix (Number alpha, const DenseSymMatrix &A, Number beta)`
Method for adding another matrix to this one.
- `void HighRankUpdate (bool trans, Number alpha, const DenseGenMatrix &V, Number beta)`
Method for adding a high-rank update to this matrix.
- `void HighRankUpdateTranspose (Number alpha, const MultiVectorMatrix &V1, const MultiVectorMatrix &V2, Number beta)`
Method for adding a high-rank update to this matrix.
- `void SpecialAddForLSR1 (const DenseVector &D, const DenseGenMatrix &L)`
Method for doing a specialized Add operation, required in the limited memory SR1 update.

Constructors / Destructors

- `DenseSymMatrix (const DenseSymMatrixSpace *owner_space)`
Constructor, taking the owner_space.
- `~DenseSymMatrix ()`
Destructor.

Protected Member Functions

Overloaded methods from Matrix base class

- `virtual void MultVectorImpl (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix-vector multiply.
- `virtual bool HasValidNumbersImpl () const`
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- `virtual void ComputeRowAMaxImpl (Vector &rows_norms, bool init) const`
Compute the max-norm of the rows in the matrix.
- `virtual void PrintImpl (const Journalist &jnlst, EJournalLevel level, EJournalCategory category, const std::string &name, Index indent, const std::string &prefix) const`
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `DenseSymMatrix ()`
Default Constructor.
- `DenseSymMatrix (const DenseSymMatrix &)`
Copy Constructor.
- `void operator= (const DenseSymMatrix &)`
Overloaded Equals Operator.

Private Attributes

- const [DenseSymMatrixSpace](#) * [owner_space_](#)
- [Number](#) * [values_](#)
Array for storing the matrix elements (one columns after each other)
- bool [initialized_](#)
Flag indicating whether the [values_](#) array has been initialized.

Additional Inherited Members

6.28.1 Detailed Description

Class for dense symetrix matrices.

[Matrix](#) elements are stored in one array in "Fortran" format, using BLAS "lower triangular" storage (not packed).

Definition at line 31 of file IpDenseSymMatrix.hpp.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Ipopt::DenseSymMatrix::DenseSymMatrix (const DenseSymMatrixSpace * owner_space)

Constructor, taking the owner_space.

6.28.2.2 Ipopt::DenseSymMatrix::~~DenseSymMatrix ()

Destructor.

6.28.2.3 Ipopt::DenseSymMatrix::DenseSymMatrix () [private]

Default Constructor.

6.28.2.4 Ipopt::DenseSymMatrix::DenseSymMatrix (const DenseSymMatrix &) [private]

Copy Constructor.

6.28.3 Member Function Documentation

6.28.3.1 SmartPtr< DenseSymMatrix > Ipopt::DenseSymMatrix::MakeNewDenseSymMatrix () const [inline]

Create a new [DenseSymMatrix](#) from same [MatrixSpace](#).

Definition at line 180 of file IpDenseSymMatrix.hpp.

6.28.3.2 Number* Ipopt::DenseSymMatrix::Values () [inline]

Retrieve the array for storing the matrix elements.

This is the non-const version, and it is assume that afterwards the calling method will set all matrix elements. The matrix elements are stored one column after each other.

Definition at line 53 of file IpDenseSymMatrix.hpp.

6.28.3.3 `const Number* Ipopt::DenseSymMatrix::Values () const [inline]`

Retrieve the array that stores the matrix elements.

This is the const version, i.e., read-only. The matrix elements are stored one column after each other.

Definition at line 63 of file IpDenseSymMatrix.hpp.

6.28.3.4 `void Ipopt::DenseSymMatrix::FillIdentity (Number factor = 1 .)`

Set this matrix to be a multiple of the identity matrix.

6.28.3.5 `void Ipopt::DenseSymMatrix::AddMatrix (Number alpha, const DenseSymMatrix & A, Number beta)`

Method for adding another matrix to this one.

If B is this matrix, it becomes $B = \alpha * A + \beta * B$ after this call.

6.28.3.6 `void Ipopt::DenseSymMatrix::HighRankUpdate (bool trans, Number alpha, const DenseGenMatrix & V, Number beta)`

Method for adding a high-rank update to this matrix.

It computes $M = \alpha * \text{op}(V) \text{op}(V)^T + \beta * M$, where V is a [DenseGenMatrix](#), where $\text{op}(V)$ is V^T if trans is true.

6.28.3.7 `void Ipopt::DenseSymMatrix::HighRankUpdateTranspose (Number alpha, const MultiVectorMatrix & V1, const MultiVectorMatrix & V2, Number beta)`

Method for adding a high-rank update to this matrix.

It computes $M = \alpha * V1^T V2 + \beta * M$, where V1 and V2 are MultiVectorMatrices, so that $V1^T V2$ is symmetric.

6.28.3.8 `void Ipopt::DenseSymMatrix::SpecialAddForLMSR1 (const DenseVector & D, const DenseGenMatrix & L)`

Method for doing a specialized Add operation, required in the limited memory SR1 update.

If M is this matrix, it computes $M = M + D + L + L^T$, where D is a diagonal matrix (given as a [DenseVector](#)), and L is a matrix that is assumed to be strictly lower triangular.

6.28.3.9 `virtual void Ipopt::DenseSymMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.28.3.10 `virtual bool Ipopt::DenseSymMatrix::IsValidNumbersImpl () const [protected], [virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.28.3.11 `virtual void Ipopt::DenseSymMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const [protected], [virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.28.3.12 `virtual void Ipopt::DenseSymMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected],[virtual]

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.28.3.13 `void Ipopt::DenseSymMatrix::operator= (const DenseSymMatrix &)` [private]

Overloaded Equals Operator.

6.28.4 Member Data Documentation

6.28.4.1 `const DenseSymMatrixSpace* Ipopt::DenseSymMatrix::owner_space_` [private]

Definition at line 137 of file `IpDenseSymMatrix.hpp`.

6.28.4.2 `Number* Ipopt::DenseSymMatrix::values_` [private]

Array for storing the matrix elements (one columns after each other)

Definition at line 141 of file `IpDenseSymMatrix.hpp`.

6.28.4.3 `bool Ipopt::DenseSymMatrix::initialized_` [private]

Flag indicating whether the `values_` array has been initialized.

Definition at line 144 of file `IpDenseSymMatrix.hpp`.

The documentation for this class was generated from the following file:

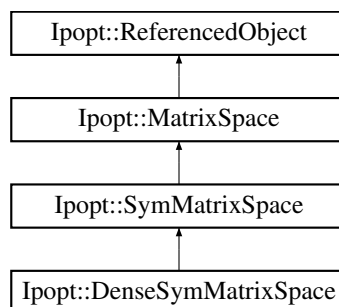
- [LinAlg/IpDenseSymMatrix.hpp](#)

6.29 Ipopt::DenseSymMatrixSpace Class Reference

This is the matrix space for [DenseSymMatrix](#).

```
#include <IpDenseSymMatrix.hpp>
```

Inheritance diagram for `Ipopt::DenseSymMatrixSpace`:



Public Member Functions

- [DenseSymMatrix](#) * [MakeNewDenseSymMatrix](#) () const

Method for creating a new matrix of this specific type.

- virtual [SymMatrix](#) * [MakeNewSymMatrix](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.

Constructors / Destructors

- [DenseSymMatrixSpace](#) (Index nDim)
Constructor for matrix space for DenseSymMatrices.
- [~DenseSymMatrixSpace](#) ()
Destructor.

6.29.1 Detailed Description

This is the matrix space for [DenseSymMatrix](#).

Definition at line 149 of file [IpDenseSymMatrix.hpp](#).

6.29.2 Constructor & Destructor Documentation

6.29.2.1 [Ipopt::DenseSymMatrixSpace::DenseSymMatrixSpace](#) (Index nDim)

Constructor for matrix space for DenseSymMatrices.

Takes in dimension of the matrices.

6.29.2.2 [Ipopt::DenseSymMatrixSpace::~~DenseSymMatrixSpace](#) () [inline]

Destructor.

Definition at line 160 of file [IpDenseSymMatrix.hpp](#).

6.29.3 Member Function Documentation

6.29.3.1 [DenseSymMatrix*](#) [Ipopt::DenseSymMatrixSpace::MakeNewDenseSymMatrix](#) () const [inline]

Method for creating a new matrix of this specific type.

Definition at line 165 of file [IpDenseSymMatrix.hpp](#).

6.29.3.2 virtual [SymMatrix*](#) [Ipopt::DenseSymMatrixSpace::MakeNewSymMatrix](#) () const [inline],[virtual]

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::SymMatrixSpace](#).

Definition at line 172 of file [IpDenseSymMatrix.hpp](#).

The documentation for this class was generated from the following file:

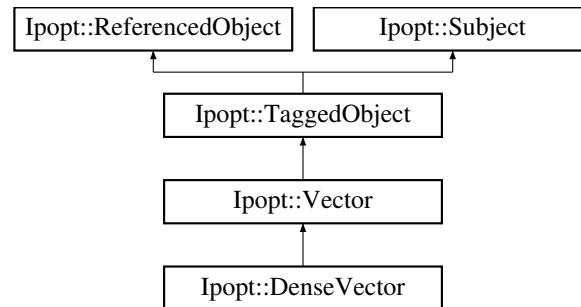
- [LinAlg/IpDenseSymMatrix.hpp](#)

6.30 [Ipopt::DenseVector](#) Class Reference

Dense [Vector](#) Implementation.


```
#include <IpDenseVector.hpp>
```

Inheritance diagram for Ipopt::DenseVector:



Public Member Functions

Constructors / Destructors

- [DenseVector](#) (const [DenseVectorSpace](#) *owner_space)
Default Constructor.
- virtual [~DenseVector](#) ()
Destructor.

Additional public methods not in Vector base class.

- [SmartPtr< DenseVector > MakeNewDenseVector](#) () const
Create a new [DenseVector](#) from same [VectorSpace](#).
- void [SetValues](#) (const [Number](#) *x)
Set elements in the vector to the [Number](#) array x.
- [Number](#) * [Values](#) ()
Obtain pointer to the internal [Number](#) array with vector elements with the intention to change the vector data (USE WITH CARE!).
- const [Number](#) * [Values](#) () const
Obtain pointer to the internal [Number](#) array with vector elements without the intention to change the vector data (USE WITH CARE!).
- const [Number](#) * [ExpandedValues](#) () const
The same as the const version of [Values](#), but we ensure that we always return a valid array, even if [IsHomogeneous](#) returns true.
- [Number](#) * [ExpandedValues](#) ()
This is the same as [Values](#), but we add it here so that [ExpandedValues](#) can also be used for the non-const case.
- bool [IsHomogeneous](#) () const
Indicates if the vector is homogeneous (i.e., all entries have the value [Scalar\(\)](#))
- [Number](#) [Scalar](#) () const
Scalar value of all entries in a homogeneous vector.

Modifying subranges of the vector.

- void [CopyToPos](#) ([Index](#) Pos, const [Vector](#) &x)
Copy the data in x into the subrange of this vector starting at position Pos in this vector.
- void [CopyFromPos](#) ([Index](#) Pos, const [Vector](#) &x)
Copy a subrange of x, starting at Pos, into the full data of this vector.

Protected Member Functions

Overloaded methods from Vector base class

- virtual void `CopyImpl` (const `Vector` &x)
Copy the data of the vector x into this vector (DCOPY).
- virtual void `ScallImpl` (`Number` alpha)
Scales the vector by scalar alpha (DSCAL)
- virtual void `AxpyImpl` (`Number` alpha, const `Vector` &x)
Add the multiple alpha of vector x to this vector (DAXPY)
- virtual `Number DotImpl` (const `Vector` &x) const
Computes inner product of vector x with this (DDOT)
- virtual `Number Nrm2Impl` () const
Computes the 2-norm of this vector (DNRM2)
- virtual `Number AsumImpl` () const
Computes the 1-norm of this vector (DASUM)
- virtual `Number AmaxImpl` () const
Computes the max-norm of this vector (based on IDAMAX)
- virtual void `SetImpl` (`Number` value)
Set each element in the vector to the scalar alpha.
- virtual void `ElementWiseDivideImpl` (const `Vector` &x)
Element-wise division $y_i \leftarrow y_i / x_i$.
- virtual void `ElementWiseMultiplyImpl` (const `Vector` &x)
*Element-wise multiplication $y_i \leftarrow y_i * x_i$.*
- virtual void `ElementWiseMaxImpl` (const `Vector` &x)
Set entry to max of itself and the corresponding element in x.
- virtual void `ElementWiseMinImpl` (const `Vector` &x)
Set entry to min of itself and the corresponding element in x.
- virtual void `ElementWiseReciprocalImpl` ()
reciprocates the elements of the vector
- virtual void `ElementWiseAbsImpl` ()
take abs of the elements of the vector
- virtual void `ElementWiseSqrtImpl` ()
take square-root of the elements of the vector
- virtual void `ElementWiseSgnImpl` ()
Changes each entry in the vector to its sgn value.
- virtual void `AddScalarImpl` (`Number` scalar)
Add scalar to every component of the vector.
- virtual `Number MaxImpl` () const
Max value in the vector.
- virtual `Number MinImpl` () const
Min value in the vector.
- virtual `Number SumImpl` () const
Computes the sum of the lements of vector.
- virtual `Number SumLogsImpl` () const
Computes the sum of the logs of the elements of vector.

Implemented specialized functions

- void `AddTwoVectorsImpl` (`Number` a, const `Vector` &v1, `Number` b, const `Vector` &v2, `Number` c)
*Add two vectors ($a * v1 + b * v2$).*
- `Number FracToBoundImpl` (const `Vector` &delta, `Number` tau) const
Fraction to the boundary parameter.
- void `AddVectorQuotientImpl` (`Number` a, const `Vector` &z, const `Vector` &s, `Number` c)
*Add the quotient of two vectors, $y = a * z/s + c * y$.*

Output methods

- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print the entire vector.
- void [PrintImplOffset](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix, [Index](#) offset) const

Private Member Functions

- [Number](#) * [values_allocated](#) ()
Method of getting the internal values array, making sure that memory has been allocated.
- void [set_values_from_scalar](#) ()
Auxilliary method for setting explicitly all elements in values_ to the current scalar value.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [DenseVector](#) ()
Default Constructor.
- [DenseVector](#) (const [DenseVector](#) &)
Copy Constructor.
- void [operator=](#) (const [DenseVector](#) &)
Overloaded Equals Operator.

Private Attributes

- const [DenseVectorSpace](#) * [owner_space_](#)
Copy of the owner_space ptr as a [DenseVectorSpace](#) instead of a [VectorSpace](#).
- [Number](#) * [values_](#)
Dense Number array of vector values.
- [Number](#) * [expanded_values_](#)
Dense Number array pointer that is used for ExpandedValues.
- bool [initialized_](#)
Flag for Initialization.
- bool [homogeneous_](#)
Flag indicating whether the vector is currently homogeneous (that is, all elements have the same value).
- [Number](#) [scalar_](#)
Homogeneous value of all elements if the vector is currently homogenous.

Friends

- class [ParVector](#)

Additional Inherited Members

6.30.1 Detailed Description

Dense [Vector](#) Implementation.

This is the default [Vector](#) class in [Ipopt](#). It stores vectors in contiguous Number arrays, unless the vector has the same value in all entries. In the latter case, we call the vector "homogeneous", and we store only the values that is repeated in all elements. If you want to obtain the values of vector, use the [IsHomogeneous\(\)](#) method to find out what status the vector is in, and then use either [Values\(\) const](#) or [Scalar\(\) const](#) methods to get the values. To set the values of a homogeneous method, use the Set method. To set the values of a non-homogeneous vector, use the SetValues method, or use the non-const Values method to get an array that you can overwrite. In the latter case, storage is ensured.

Definition at line 40 of file IpDenseVector.hpp.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 `Ipopt::DenseVector::DenseVector (const DenseVectorSpace * owner_space)`

Default Constructor.

6.30.2.2 `virtual Ipopt::DenseVector::~DenseVector () [virtual]`

Destructor.

6.30.2.3 `Ipopt::DenseVector::DenseVector () [private]`

Default Constructor.

6.30.2.4 `Ipopt::DenseVector::DenseVector (const DenseVector &) [private]`

Copy Constructor.

6.30.3 Member Function Documentation

6.30.3.1 `SmartPointer< DenseVector > Ipopt::DenseVector::MakeNewDenseVector () const [inline]`

Create a new [DenseVector](#) from same [VectorSpace](#).

Definition at line 442 of file IpDenseVector.hpp.

6.30.3.2 `void Ipopt::DenseVector::SetValues (const Number * x)`

Set elements in the vector to the Number array x.

6.30.3.3 `Number * Ipopt::DenseVector::Values () [inline]`

Obtain pointer to the internal Number array with vector elements with the intention to change the vector data (USE WITH CARE!).

This does not produce a copy, and lifetime is not guaranteed!.

Definition at line 393 of file IpDenseVector.hpp.

6.30.3.4 `const Number* Ipopt::DenseVector::Values () const [inline]`

Obtain pointer to the internal Number array with vector elements without the intention to change the vector data (USE WITH CARE!).

This does not produce a copy, and lifetime is not guaranteed! IMPORTANT: If this method is currently homogeneous (i.e. `IsHomogeneous` returns true), then you cannot call this method. Instead, you need to use the [Scalar\(\)](#) method.

Definition at line 410 of file `IpDenseVector.hpp`.

6.30.3.5 `const Number* Ipopt::DenseVector::ExpandedValues () const`

The same as the const version of `Values`, but we ensure that we always return a valid array, even if `IsHomogeneous` returns true.

6.30.3.6 `Number* Ipopt::DenseVector::ExpandedValues () [inline]`

This is the same as `Values`, but we add it here so that `ExpandedValues` can also be used for the non-const case.

Definition at line 87 of file `IpDenseVector.hpp`.

6.30.3.7 `bool Ipopt::DenseVector::IsHomogeneous () const [inline]`

Indicates if the vector is homogeneous (i.e., all entries have the value [Scalar\(\)](#))

Definition at line 94 of file `IpDenseVector.hpp`.

6.30.3.8 `Number Ipopt::DenseVector::Scalar () const [inline]`

Scalar value of all entries in a homogeneous vector.

Definition at line 100 of file `IpDenseVector.hpp`.

6.30.3.9 `void Ipopt::DenseVector::CopyToPos (Index Pos, const Vector & x)`

Copy the data in x into the subrange of this vector starting at position Pos in this vector.

Position count starts at 0.

6.30.3.10 `void Ipopt::DenseVector::CopyFromPos (Index Pos, const Vector & x)`

Copy a subrange of x, starting at Pos, into the full data of this vector.

Position count starts at 0.

6.30.3.11 `virtual void Ipopt::DenseVector::CopyImpl (const Vector & x) [protected],[virtual]`

Copy the data of the vector x into this vector (DCOPY).

Implements [Ipopt::Vector](#).

6.30.3.12 `virtual void Ipopt::DenseVector::ScalImpl (Number alpha) [protected],[virtual]`

Scales the vector by scalar alpha (DSCAL)

Implements [Ipopt::Vector](#).

6.30.3.13 `virtual void Ipopt::DenseVector::AxpymImpl (Number alpha, const Vector & x) [protected],[virtual]`

Add the multiple alpha of vector x to this vector (DAXPY)

Implements [Ipopt::Vector](#).

6.30.3.14 `virtual Number lpopt::DenseVector::DotImpl (const Vector & x) const` [protected],[virtual]

Computes inner product of vector x with this (DDOT)

Implements [lpopt::Vector](#).

6.30.3.15 `virtual Number lpopt::DenseVector::Nrm2Impl () const` [protected],[virtual]

Computes the 2-norm of this vector (DNRM2)

Implements [lpopt::Vector](#).

6.30.3.16 `virtual Number lpopt::DenseVector::AsumImpl () const` [protected],[virtual]

Computes the 1-norm of this vector (DASUM)

Implements [lpopt::Vector](#).

6.30.3.17 `virtual Number lpopt::DenseVector::AmaxImpl () const` [protected],[virtual]

Computes the max-norm of this vector (based on IDAMAX)

Implements [lpopt::Vector](#).

6.30.3.18 `virtual void lpopt::DenseVector::SetImpl (Number value)` [protected],[virtual]

Set each element in the vector to the scalar alpha.

Implements [lpopt::Vector](#).

6.30.3.19 `virtual void lpopt::DenseVector::ElementWiseDivideImpl (const Vector & x)` [protected],[virtual]

Element-wise division $y_i \leftarrow y_i/x_i$.

Implements [lpopt::Vector](#).

6.30.3.20 `virtual void lpopt::DenseVector::ElementWiseMultiplyImpl (const Vector & x)` [protected],[virtual]

Element-wise multiplication $y_i \leftarrow y_i * x_i$.

Implements [lpopt::Vector](#).

6.30.3.21 `virtual void lpopt::DenseVector::ElementWiseMaxImpl (const Vector & x)` [protected],[virtual]

Set entry to max of itself and the corresponding element in x.

Implements [lpopt::Vector](#).

6.30.3.22 `virtual void lpopt::DenseVector::ElementWiseMinImpl (const Vector & x)` [protected],[virtual]

Set entry to min of itself and the corresponding element in x.

Implements [lpopt::Vector](#).

6.30.3.23 `virtual void lpopt::DenseVector::ElementWiseReciprocalImpl ()` [protected],[virtual]

reciprocates the elements of the vector

Implements [lpopt::Vector](#).

6.30.3.24 `virtual void Ipopt::DenseVector::ElementWiseAbsImpl () [protected], [virtual]`

take abs of the elements of the vector

Implements [Ipopt::Vector](#).

6.30.3.25 `virtual void Ipopt::DenseVector::ElementWiseSqrtImpl () [protected], [virtual]`

take square-root of the elements of the vector

Implements [Ipopt::Vector](#).

6.30.3.26 `virtual void Ipopt::DenseVector::ElementWiseSgnImpl () [protected], [virtual]`

Changes each entry in the vector to its sgn value.

Implements [Ipopt::Vector](#).

6.30.3.27 `virtual void Ipopt::DenseVector::AddScalarImpl (Number scalar) [protected], [virtual]`

Add scalar to every component of the vector.

Implements [Ipopt::Vector](#).

6.30.3.28 `virtual Number Ipopt::DenseVector::MaxImpl () const [protected], [virtual]`

Max value in the vector.

Implements [Ipopt::Vector](#).

6.30.3.29 `virtual Number Ipopt::DenseVector::MinImpl () const [protected], [virtual]`

Min value in the vector.

Implements [Ipopt::Vector](#).

6.30.3.30 `virtual Number Ipopt::DenseVector::SumImpl () const [protected], [virtual]`

Computes the sum of the lements of vector.

Implements [Ipopt::Vector](#).

6.30.3.31 `virtual Number Ipopt::DenseVector::SumLogsImpl () const [protected], [virtual]`

Computes the sum of the logs of the elements of vector.

Implements [Ipopt::Vector](#).

6.30.3.32 `void Ipopt::DenseVector::AddTwoVectorsImpl (Number a, const Vector & v1, Number b, const Vector & v2, Number c) [protected], [virtual]`

Add two vectors ($a * v1 + b * v2$).

Result is stored in this vector.

Reimplemented from [Ipopt::Vector](#).

6.30.3.33 `Number Ipopt::DenseVector::FracToBoundImpl (const Vector & delta, Number tau) const [protected], [virtual]`

Fraction to the boundary parameter.

Reimplemented from [Ipopt::Vector](#).

```
6.30.3.34 void Ipopt::DenseVector::AddVectorQuotientImpl ( Number a, const Vector & z, const Vector & s, Number c )
           [protected], [virtual]
```

Add the quotient of two vectors, $y = a * z/s + c * y$.

Reimplemented from [Ipopt::Vector](#).

```
6.30.3.35 virtual void Ipopt::DenseVector::PrintImpl ( const Journalist & jnlst, EJournalLevel level, EJournalCategory
           category, const std::string & name, Index indent, const std::string & prefix ) const [inline], [protected],
           [virtual]
```

Print the entire vector.

Implements [Ipopt::Vector](#).

Definition at line 201 of file IpDenseVector.hpp.

```
6.30.3.36 void Ipopt::DenseVector::PrintImplOffset ( const Journalist & jnlst, EJournalLevel level, EJournalCategory
           category, const std::string & name, Index indent, const std::string & prefix, Index offset ) const [protected]
```

```
6.30.3.37 void Ipopt::DenseVector::operator= ( const DenseVector & ) [private]
```

Overloaded Equals Operator.

```
6.30.3.38 Number * Ipopt::DenseVector::values_allocated ( ) [inline], [private]
```

Method of getting the internal values array, making sure that memory has been allocated.

Definition at line 416 of file IpDenseVector.hpp.

```
6.30.3.39 void Ipopt::DenseVector::set_values_from_scalar ( ) [private]
```

Auxilliary method for setting explicitly all elements in values_ to the current scalar value.

6.30.4 Friends And Related Function Documentation

```
6.30.4.1 friend class ParVector [friend]
```

Definition at line 220 of file IpDenseVector.hpp.

6.30.5 Member Data Documentation

```
6.30.5.1 const DenseVectorSpace* Ipopt::DenseVector::owner_space_ [private]
```

Copy of the owner_space ptr as a [DenseVectorSpace](#) instead of a [VectorSpace](#).

Definition at line 244 of file IpDenseVector.hpp.

```
6.30.5.2 Number* Ipopt::DenseVector::values_ [private]
```

Dense Number array of vector values.

Definition at line 247 of file IpDenseVector.hpp.

6.30.5.3 `Number*` `Ipopt::DenseVector::expanded_values_` `[mutable]`, `[private]`

Dense Number array pointer that is used for ExpandedValues.

Definition at line 250 of file `IpDenseVector.hpp`.

6.30.5.4 `bool` `Ipopt::DenseVector::initialized_` `[private]`

Flag for Initialization.

This flag is false, if the data has not yet been initialized.

Definition at line 259 of file `IpDenseVector.hpp`.

6.30.5.5 `bool` `Ipopt::DenseVector::homogeneous_` `[private]`

Flag indicating whether the vector is currently homogeneous (that is, all elements have the same value).

This flag is used to determine whether the elements of the vector are stored in `values_` or in `scalar_`

Definition at line 265 of file `IpDenseVector.hpp`.

6.30.5.6 `Number` `Ipopt::DenseVector::scalar_` `[private]`

Homogeneous value of all elements if the vector is currently homogenous.

Definition at line 269 of file `IpDenseVector.hpp`.

The documentation for this class was generated from the following file:

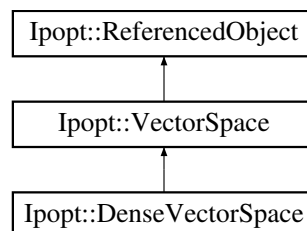
- [LinAlg/IpDenseVector.hpp](#)

6.31 Ipopt::DenseVectorSpace Class Reference

This vectors space is the vector space for [DenseVector](#).

```
#include <IpDenseVector.hpp>
```

Inheritance diagram for `Ipopt::DenseVectorSpace`:



Public Member Functions

- [DenseVector](#) * [MakeNewDenseVector](#) () const
Method for creating a new vector of this specific type.
- virtual [Vector](#) * [MakeNew](#) () const
Instantiation of the generate MakeNew method for the [VectorSpace](#) base class.

Constructors/Destructors.

- [DenseVectorSpace](#) ([Index](#) dim)
Constructor, requires dimension of all vector for this [VectorSpace](#).
- [~DenseVectorSpace](#) ()
Destructor.

Methods called by DenseVector for memory management.

This could allow to have sophisticated memory management in the [VectorSpace](#).

- [Number](#) * [AllocateInternalStorage](#) () const
Allocate internal storage for the [DenseVector](#).
- void [FreeInternalStorage](#) ([Number](#) *values) const
Deallocate internal storage for the [DenseVector](#).

Methods for dealing with meta data on the vector

- bool [HasStringMetaData](#) (const std::string tag) const
Check if string meta exists for tag.
- bool [HasIntegerMetaData](#) (const std::string tag) const
Check if Integer meta exists for tag.
- bool [HasNumericMetaData](#) (const std::string tag) const
Check if Numeric meta exists for tag.
- const std::vector< std::string > & [GetStringMetaData](#) (const std::string &tag) const
Get meta data of type std::string by tag.
- const std::vector< [Index](#) > & [GetIntegerMetaData](#) (const std::string &tag) const
Get meta data of type Index by tag.
- const std::vector< [Number](#) > & [GetNumericMetaData](#) (const std::string &tag) const
Get meta data of type Number by tag.
- void [SetStringMetaData](#) (std::string tag, std::vector< std::string > meta_data)
Set meta data of type std::string by tag.
- void [SetIntegerMetaData](#) (std::string tag, std::vector< [Index](#) > meta_data)
Set meta data of type Index by tag.
- void [SetNumericMetaData](#) (std::string tag, std::vector< [Number](#) > meta_data)
Set meta data of type Number by tag.
- const [StringMetaDataMapType](#) & [GetStringMetaData](#) () const
Get map of meta data of type Number.
- const [IntegerMetaDataMapType](#) & [GetIntegerMetaData](#) () const
Get map of meta data of type Number.
- const [NumericMetaDataMapType](#) & [GetNumericMetaData](#) () const
Get map of meta data of type Number.

Private Attributes

- [StringMetaDataMapType](#) string_meta_data_
- [IntegerMetaDataMapType](#) integer_meta_data_
- [NumericMetaDataMapType](#) numeric_meta_data_

6.31.1 Detailed Description

This vectors space is the vector space for [DenseVector](#).

Definition at line 285 of file [lpDenseVector.hpp](#).

6.31.2 Constructor & Destructor Documentation

6.31.2.1 Ipopt::DenseVectorSpace::DenseVectorSpace (Index *dim*) [inline]

Constructor, requires dimension of all vector for this [VectorSpace](#).

Definition at line 293 of file IpDenseVector.hpp.

6.31.2.2 Ipopt::DenseVectorSpace::~~DenseVectorSpace () [inline]

Destructor.

Definition at line 299 of file IpDenseVector.hpp.

6.31.3 Member Function Documentation

6.31.3.1 DenseVector* Ipopt::DenseVectorSpace::MakeNewDenseVector () const [inline]

Method for creating a new vector of this specific type.

Definition at line 305 of file IpDenseVector.hpp.

6.31.3.2 virtual Vector* Ipopt::DenseVectorSpace::MakeNew () const [inline],[virtual]

Instantiation of the generate MakeNew method for the [VectorSpace](#) base class.

Implements [Ipopt::VectorSpace](#).

Definition at line 313 of file IpDenseVector.hpp.

6.31.3.3 Number * Ipopt::DenseVectorSpace::AllocateInternalStorage () const [inline]

Allocate internal storage for the [DenseVector](#).

Definition at line 425 of file IpDenseVector.hpp.

6.31.3.4 void Ipopt::DenseVectorSpace::FreeInternalStorage (Number * *values*) const [inline]

Deallocate internal storage for the [DenseVector](#).

Definition at line 436 of file IpDenseVector.hpp.

6.31.3.5 bool Ipopt::DenseVectorSpace::HasStringMetaData (const std::string *tag*) const [inline]

Check if string meta exists for tag.

Definition at line 448 of file IpDenseVector.hpp.

6.31.3.6 bool Ipopt::DenseVectorSpace::HasIntegerMetaData (const std::string *tag*) const [inline]

Check if Integer meta exists for tag.

Definition at line 461 of file IpDenseVector.hpp.

6.31.3.7 bool Ipopt::DenseVectorSpace::HasNumericMetaData (const std::string *tag*) const [inline]

Check if Numeric meta exists for tag.

Definition at line 474 of file IpDenseVector.hpp.

6.31.3.8 `const std::vector< std::string > & Ipopt::DenseVectorSpace::GetStringMetaData (const std::string & tag) const`
`[inline]`

Get meta data of type `std::string` by tag.

Definition at line 487 of file `IpDenseVector.hpp`.

6.31.3.9 `const std::vector< Index > & Ipopt::DenseVectorSpace::GetIntegerMetaData (const std::string & tag) const`
`[inline]`

Get meta data of type `Index` by tag.

Definition at line 496 of file `IpDenseVector.hpp`.

6.31.3.10 `const std::vector< Number > & Ipopt::DenseVectorSpace::GetNumericMetaData (const std::string & tag) const`
`[inline]`

Get meta data of type `Number` by tag.

Definition at line 505 of file `IpDenseVector.hpp`.

6.31.3.11 `void Ipopt::DenseVectorSpace::SetStringMetaData (std::string tag, std::vector< std::string > meta_data)`
`[inline]`

Set meta data of type `std::string` by tag.

Definition at line 514 of file `IpDenseVector.hpp`.

6.31.3.12 `void Ipopt::DenseVectorSpace::SetIntegerMetaData (std::string tag, std::vector< Index > meta_data)` `[inline]`

Set meta data of type `Index` by tag.

Definition at line 520 of file `IpDenseVector.hpp`.

6.31.3.13 `void Ipopt::DenseVectorSpace::SetNumericMetaData (std::string tag, std::vector< Number > meta_data)`
`[inline]`

Set meta data of type `Number` by tag.

Definition at line 526 of file `IpDenseVector.hpp`.

6.31.3.14 `const StringMetaDataMapType & Ipopt::DenseVectorSpace::GetStringMetaData () const` `[inline]`

Get map of meta data of type `Number`.

Definition at line 532 of file `IpDenseVector.hpp`.

6.31.3.15 `const IntegerMetaDataMapType & Ipopt::DenseVectorSpace::GetIntegerMetaData () const` `[inline]`

Get map of meta data of type `Number`.

Definition at line 538 of file `IpDenseVector.hpp`.

6.31.3.16 `const NumericMetaDataMapType & Ipopt::DenseVectorSpace::GetNumericMetaData () const` `[inline]`

Get map of meta data of type `Number`.

Definition at line 544 of file `IpDenseVector.hpp`.

6.31.4 Member Data Documentation

6.31.4.1 StringMetaDataMapType Ipopt::DenseVectorSpace::string_meta_data_ [private]

Definition at line 386 of file IpDenseVector.hpp.

6.31.4.2 IntegerMetaDataMapType Ipopt::DenseVectorSpace::integer_meta_data_ [private]

Definition at line 387 of file IpDenseVector.hpp.

6.31.4.3 NumericMetaDataMapType Ipopt::DenseVectorSpace::numeric_meta_data_ [private]

Definition at line 388 of file IpDenseVector.hpp.

The documentation for this class was generated from the following file:

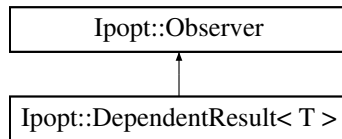
- [LinAlg/IpDenseVector.hpp](#)

6.32 Ipopt::DependentResult< T > Class Template Reference

Templated class which stores one entry for the CachedResult class.

```
#include <IpCachedResults.hpp>
```

Inheritance diagram for Ipopt::DependentResult< T >:



Public Member Functions

- bool [DependentsIdentical](#) (const std::vector< const [TaggedObject](#) * > &dependents, const std::vector< [Number](#) > &scalar_dependents) const
This method returns true if the dependencies provided to this function are identical to the ones stored with the [DependentResult](#).
- void [DebugPrint](#) () const
Print information about this [DependentResults](#).

Constructor, Destructors

- [DependentResult](#) (const T &result, const std::vector< const [TaggedObject](#) * > &dependents, const std::vector< [Number](#) > &scalar_dependents)
Constructor, given all information about the result.
- [~DependentResult](#) ()
Destructor.

Accessor method.

- bool [IsStale](#) () const
This returns true, if the [DependentResult](#) is no longer valid.
- void [Invalidate](#) ()
Invalidates the cached result.
- const T & [GetResult](#) () const
Returns the cached result.

Protected Member Functions

- virtual void [RecieveNotification](#) ([NotifyType](#) notify_type, const [Subject](#) *subject)

This method is overloading the pure virtual method from the [Observer](#) base class.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [DependentResult](#) ()
Default Constructor.
- [DependentResult](#) (const [DependentResult](#) &)
Copy Constructor.
- void [operator=](#) (const [DependentResult](#) &)
Overloaded Equals Operator.

Private Attributes

- bool [stale_](#)
Flag indicating, if the cached result is still valid.
- const T [result_](#)
The value of the dependent results.
- std::vector< [TaggedObject::Tag](#) > [dependent_tags_](#)
Dependencies in form of TaggedObjects.
- std::vector< [Number](#) > [scalar_dependents_](#)
Dependencies in form a Numbers.

Additional Inherited Members

6.32.1 Detailed Description

```
template<class T>class Ipopt::DependentResult< T >
```

Templated class which stores one entry for the CachedResult class.

It stores the result (of type T), together with its dependencies (vector of TaggedObjects and vector of Numbers). It also stores a priority.

Definition at line 31 of file IpCachedResults.hpp.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 `template<class T> Ipopt::DependentResult< T >::DependentResult (const T & result, const std::vector< const TaggedObject * > & dependents, const std::vector< Number > & scalar_dependents)`

Constructor, given all information about the result.

Definition at line 348 of file IpCachedResults.hpp.

6.32.2.2 `template<class T> Ipopt::DependentResult< T >::~~DependentResult ()`

Destructor.

Definition at line 381 of file IpCachedResults.hpp.

6.32.2.3 `template<class T> Ipopt::DependentResult< T >::DependentResult () [private]`

Default Constructor.

6.32.2.4 `template<class T> Ipopt::DependentResult< T >::DependentResult (const DependentResult< T > &) [private]`

Copy Constructor.

6.32.3 Member Function Documentation

6.32.3.1 `template<class T> bool Ipopt::DependentResult< T >::IsStale () const`

This returns true, if the [DependentResult](#) is no longer valid.

Definition at line 393 of file IpCachedResults.hpp.

6.32.3.2 `template<class T> void Ipopt::DependentResult< T >::Invalidate ()`

Invalidates the cached result.

Definition at line 399 of file IpCachedResults.hpp.

6.32.3.3 `template<class T> const T & Ipopt::DependentResult< T >::GetResult () const`

Returns the cached result.

Definition at line 456 of file IpCachedResults.hpp.

6.32.3.4 `template<class T> bool Ipopt::DependentResult< T >::DependentsIdentical (const std::vector< const TaggedObject * > & depends, const std::vector< Number > & scalar_depends) const`

This method returns true if the dependencies provided to this function are identical to the ones stored with the [DependentResult](#).

Definition at line 419 of file IpCachedResults.hpp.

6.32.3.5 `template<class T> void Ipopt::DependentResult< T >::DebugPrint () const`

Print information about this DependentResults.

Definition at line 467 of file IpCachedResults.hpp.

6.32.3.6 `template<class T> void Ipopt::DependentResult< T >::RecieveNotification (NotifyType notify_type, const Subject * subject) [protected], [virtual]`

This method is overloading the pure virtual method from the [Observer](#) base class.

This method is called when a [Subject](#) registered for this [Observer](#) sends a notification. In this particular case, if this method is called with notify_type==NT_Changed or NT_BeingDeleted, then this results is marked as stale.

Implements [Ipopt::Observer](#).

Definition at line 405 of file IpCachedResults.hpp.

6.32.3.7 `template<class T> void Ipopt::DependentResult< T >::operator= (const DependentResult< T > &)`
`[private]`

Overloaded Equals Operator.

6.32.4 Member Data Documentation

6.32.4.1 `template<class T> bool Ipopt::DependentResult< T >::stale_` `[private]`

Flag indicating, if the cached result is still valid.

A result becomes invalid, if the RecieveNotification method is called with NT_Changed

Definition at line 330 of file IpCachedResults.hpp.

6.32.4.2 `template<class T> const T Ipopt::DependentResult< T >::result_` `[private]`

The value of the dependent results.

Definition at line 332 of file IpCachedResults.hpp.

6.32.4.3 `template<class T> std::vector<TaggedObject::Tag> Ipopt::DependentResult< T >::dependent_tags_`
`[private]`

Dependencies in form of TaggedObjects.

Definition at line 334 of file IpCachedResults.hpp.

6.32.4.4 `template<class T> std::vector<Number> Ipopt::DependentResult< T >::scalar_dependents_` `[private]`

Dependencies in form a Numbers.

Definition at line 336 of file IpCachedResults.hpp.

The documentation for this class was generated from the following file:

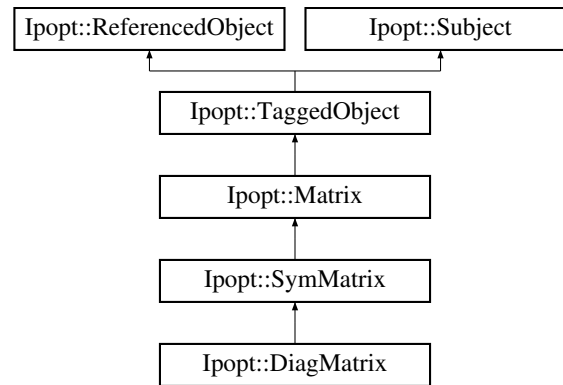
- [Common/IpCachedResults.hpp](#)

6.33 Ipopt::DiagMatrix Class Reference

Class for diagonal matrices.

```
#include <IpDiagMatrix.hpp>
```

Inheritance diagram for Ipopt::DiagMatrix:



Public Member Functions

- void [SetDiag](#) (const [Vector](#) &diag)
Method for setting the diagonal elements (as a [Vector](#)).
- [SmartPtr](#)< const [Vector](#) > [GetDiag](#) () const
Method for setting the diagonal elements.

Constructors / Destructors

- [DiagMatrix](#) (const [SymMatrixSpace](#) *owner_space)
Constructor, given the corresponding matrix space.
- [~DiagMatrix](#) ()
Destructor.

Protected Member Functions

Methods overloaded from matrix

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual bool [IsValidNumbersImpl](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [DiagMatrix](#) ()
Default Constructor.
- [DiagMatrix](#) (const [DiagMatrix](#) &)

- *Copy Constructor.*
- void `operator=` (const `DiagMatrix` &)
Overloaded Equals Operator.

Private Attributes

- `SmartPtr`< const `Vector` > `diag_`
`Vector` storing the diagonal elements.

Additional Inherited Members

6.33.1 Detailed Description

Class for diagonal matrices.

The diagonal is stored as a `Vector`.

Definition at line 20 of file `IpDiagMatrix.hpp`.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 `Ipopt::DiagMatrix::DiagMatrix (const SymMatrixSpace * owner_space)`

Constructor, given the corresponding matrix space.

6.33.2.2 `Ipopt::DiagMatrix::~~DiagMatrix ()`

Destructor.

6.33.2.3 `Ipopt::DiagMatrix::DiagMatrix ()` [private]

Default Constructor.

6.33.2.4 `Ipopt::DiagMatrix::DiagMatrix (const DiagMatrix &)` [private]

Copy Constructor.

6.33.3 Member Function Documentation

6.33.3.1 `void Ipopt::DiagMatrix::SetDiag (const Vector & diag)` [inline]

Method for setting the diagonal elements (as a `Vector`).

Definition at line 35 of file `IpDiagMatrix.hpp`.

6.33.3.2 `SmartPtr<const Vector> Ipopt::DiagMatrix::GetDiag () const` [inline]

Method for setting the diagonal elements.

Definition at line 41 of file `IpDiagMatrix.hpp`.

6.33.3.3 `virtual void Ipopt::DiagMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const` [protected], [virtual]

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.33.3.4 `virtual bool Ipopt::DiagMatrix::IsValidNumbersImpl () const` [protected],[virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.33.3.5 `virtual void Ipopt::DiagMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` [protected],[virtual]

Compute the max-norm of the rows in the matrix.

The result is stored in `rows_norms`. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.33.3.6 `virtual void Ipopt::DiagMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected],[virtual]

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.33.3.7 `void Ipopt::DiagMatrix::operator= (const DiagMatrix &)` [private]

Overloaded Equals Operator.

6.33.4 Member Data Documentation

6.33.4.1 `SmartPtr<const Vector> Ipopt::DiagMatrix::diag_` [private]

[Vector](#) storing the diagonal elements.

Definition at line 86 of file `IpDiagMatrix.hpp`.

The documentation for this class was generated from the following file:

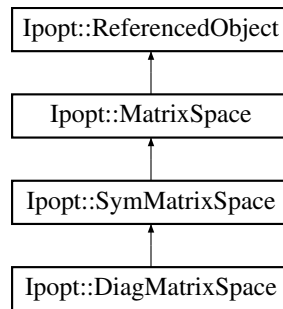
- [LinAlg/IpDiagMatrix.hpp](#)

6.34 Ipopt::DiagMatrixSpace Class Reference

This is the matrix space for [DiagMatrix](#).

```
#include <IpDiagMatrix.hpp>
```

Inheritance diagram for `Ipopt::DiagMatrixSpace`:



Public Member Functions

- virtual [SymMatrix](#) * [MakeNewSymMatrix](#) () const
Overloaded MakeNew method for the [SymMatrixSpace](#) base class.
- [DiagMatrix](#) * [MakeNewDiagMatrix](#) () const
Method for creating a new matrix of this specific type.

Constructors / Destructors

- [DiagMatrixSpace](#) ([Index](#) dim)
Constructor, given the dimension of the matrix.
- virtual [~DiagMatrixSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [DiagMatrixSpace](#) ()
Default Constructor.
- [DiagMatrixSpace](#) (const [DiagMatrixSpace](#) &)
Copy Constructor.
- void [operator=](#) (const [DiagMatrixSpace](#) &)
Overloaded Equals Operator.

6.34.1 Detailed Description

This is the matrix space for [DiagMatrix](#).

Definition at line 90 of file [IpDiagMatrix.hpp](#).

6.34.2 Constructor & Destructor Documentation

6.34.2.1 [Ipopt::DiagMatrixSpace::DiagMatrixSpace](#) ([Index](#) dim) [inline]

Constructor, given the dimension of the matrix.

Definition at line 96 of file [IpDiagMatrix.hpp](#).

6.34.2.2 `virtual Ipopt::DiagMatrixSpace::~~DiagMatrixSpace () [inline],[virtual]`

Destructor.

Definition at line 102 of file `IpDiagMatrix.hpp`.

6.34.2.3 `Ipopt::DiagMatrixSpace::DiagMatrixSpace () [private]`

Default Constructor.

6.34.2.4 `Ipopt::DiagMatrixSpace::DiagMatrixSpace (const DiagMatrixSpace &) [private]`

Copy Constructor.

6.34.3 Member Function Documentation

6.34.3.1 `virtual SymMatrix* Ipopt::DiagMatrixSpace::MakeNewSymMatrix () const [inline],[virtual]`

Overloaded MakeNew method for the [SymMatrixSpace](#) base class.

Implements [Ipopt::SymMatrixSpace](#).

Definition at line 108 of file `IpDiagMatrix.hpp`.

6.34.3.2 `DiagMatrix* Ipopt::DiagMatrixSpace::MakeNewDiagMatrix () const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 114 of file `IpDiagMatrix.hpp`.

6.34.3.3 `void Ipopt::DiagMatrixSpace::operator= (const DiagMatrixSpace &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

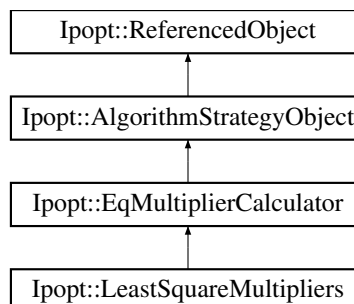
- [LinAlg/IpDiagMatrix.hpp](#)

6.35 Ipopt::EqMultiplierCalculator Class Reference

Base Class for objects that compute estimates for the equality constraint multipliers `y_c` and `y_d`.

```
#include <IpEqMultCalculator.hpp>
```

Inheritance diagram for `Ipopt::EqMultiplierCalculator`:



Public Member Functions

- virtual bool `InitializeImpl` (const `OptionsList` &options, const std::string &prefix)=0
overloaded from `AlgorithmStrategyObject`
- virtual bool `CalculateMultipliers` (`Vector` &y_c, `Vector` &y_d)=0
This method computes the estimates for y_c and y_d at the current point.

Constructors/Destructors

- `EqMultiplierCalculator` ()
Default Constructor.
- virtual `~EqMultiplierCalculator` ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `EqMultiplierCalculator` (const `EqMultiplierCalculator` &)
Copy Constructor.
- void `operator=` (const `EqMultiplierCalculator` &)
Overloaded Equals Operator.

Additional Inherited Members

6.35.1 Detailed Description

Base Class for objects that compute estimates for the equality constraint multipliers y_c and y_d.

For example, this is the base class for objects for computing least square multipliers or coordinate multipliers.

Definition at line 21 of file `IpEqMultCalculator.hpp`.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 `Ipopt::EqMultiplierCalculator::EqMultiplierCalculator ()` `[inline]`

Default Constructor.

Definition at line 27 of file `IpEqMultCalculator.hpp`.

6.35.2.2 `virtual Ipopt::EqMultiplierCalculator::~~EqMultiplierCalculator ()` `[inline]`, `[virtual]`

Default destructor.

Definition at line 30 of file `IpEqMultCalculator.hpp`.

6.35.2.3 `Ipopt::EqMultiplierCalculator::EqMultiplierCalculator (const EqMultiplierCalculator &)` `[private]`

Copy Constructor.

6.35.3 Member Function Documentation

6.35.3.1 `virtual bool Ipopt::EqMultiplierCalculator::InitializeImpl (const OptionsList & options, const std::string & prefix)`
`[pure virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::LeastSquareMultipliers](#).

6.35.3.2 `virtual bool Ipopt::EqMultiplierCalculator::CalculateMultipliers (Vector & y_c, Vector & y_d)` `[pure virtual]`

This method computes the estimates for `y_c` and `y_d` at the current point.

If the estimates cannot be computed (e.g. some linear system is singular), the return value of this method is false.

Implemented in [Ipopt::LeastSquareMultipliers](#).

6.35.3.3 `void Ipopt::EqMultiplierCalculator::operator= (const EqMultiplierCalculator &)` `[private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

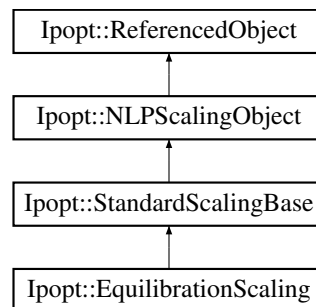
- [Algorithm/IpEqMultCalculator.hpp](#)

6.36 Ipopt::EquilibrationScaling Class Reference

This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point.

```
#include <IpEquilibrationScaling.hpp>
```

Inheritance diagram for `Ipopt::EquilibrationScaling`:



Public Member Functions

Constructors/Destructors

- [EquilibrationScaling](#) (const [SmartPtr](#)< [NLP](#) > &nlp)
- `virtual ~EquilibrationScaling ()`

Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &options)

Methods for IpoptType.

Protected Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)

Initialize the object from the options.

- virtual void [DetermineScalingParametersImpl](#) (const [SmartPtr](#)< const [VectorSpace](#) > x_space, const [SmartPtr](#)< const [VectorSpace](#) > c_space, const [SmartPtr](#)< const [VectorSpace](#) > d_space, const [SmartPtr](#)< const [MatrixSpace](#) > jac_c_space, const [SmartPtr](#)< const [MatrixSpace](#) > jac_d_space, const [SmartPtr](#)< const [SymMatrixSpace](#) > h_space, const [Matrix](#) &Px_L, const [Vector](#) &x_L, const [Matrix](#) &Px_U, const [Vector](#) &x_U, [Number](#) &df, [SmartPtr](#)< [Vector](#) > &dx, [SmartPtr](#)< [Vector](#) > &dc, [SmartPtr](#)< [Vector](#) > &dd)

This is the method that has to be overloaded by a particular scaling method that somehow computes the scaling vectors dx, dc, and dd.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [EquilibrationScaling](#) (const [EquilibrationScaling](#) &)

Copy Constructor.

- void [operator=](#) (const [EquilibrationScaling](#) &)

Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [NLP](#) > nlp_

pointer to the [NLP](#) to get scaling parameters

- [Number](#) point_perturbation_radius_

maximal radius for the random perturbation of the initial point.

6.36.1 Detailed Description

This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point.

Definition at line 21 of file IpEquilibrationScaling.hpp.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 Ipopt::EquilibrationScaling::EquilibrationScaling (const [SmartPtr](#)< [NLP](#) > & nlp) [inline]

Definition at line 26 of file IpEquilibrationScaling.hpp.

6.36.2.2 `virtual Ipopt::EquilibrationScaling::~~EquilibrationScaling () [inline],[virtual]`

Default destructor.

Definition at line 33 of file `IpEquilibrationScaling.hpp`.

6.36.2.3 `Ipopt::EquilibrationScaling::EquilibrationScaling (const EquilibrationScaling &) [private]`

Copy Constructor.

6.36.3 Member Function Documentation

6.36.3.1 `static void Ipopt::EquilibrationScaling::RegisterOptions (const SmartPtr< RegisteredOptions > & options) [static]`

Methods for IpoptType.

Register the options for this class

6.36.3.2 `bool Ipopt::EquilibrationScaling::InitializeImpl (const OptionsList & options, const std::string & prefix) [protected],[virtual]`

Initialize the object from the options.

Reimplemented from [Ipopt::StandardScalingBase](#).

6.36.3.3 `virtual void Ipopt::EquilibrationScaling::DetermineScalingParametersImpl (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, const Matrix & Px_L, const Vector & x_L, const Matrix & Px_U, const Vector & x_U, Number & df, SmartPtr< Vector > & dx, SmartPtr< Vector > & dc, SmartPtr< Vector > & dd) [protected],[virtual]`

This is the method that has to be overloaded by a particular scaling method that somehow computes the scaling vectors dx, dc, and dd.

The pointers to those vectors can be NULL, in which case no scaling for that item will be done later.

Implements [Ipopt::StandardScalingBase](#).

6.36.3.4 `void Ipopt::EquilibrationScaling::operator= (const EquilibrationScaling &) [private]`

Overloaded Equals Operator.

6.36.4 Member Data Documentation

6.36.4.1 `SmartPtr<NLP> Ipopt::EquilibrationScaling::nlp_ [private]`

pointer to the [NLP](#) to get scaling parameters

Definition at line 81 of file `IpEquilibrationScaling.hpp`.

6.36.4.2 `Number Ipopt::EquilibrationScaling::point_perturbation_radius_ [private]`

maximal radius for the random perturbation of the initial point.

Definition at line 85 of file `IpEquilibrationScaling.hpp`.

The documentation for this class was generated from the following file:

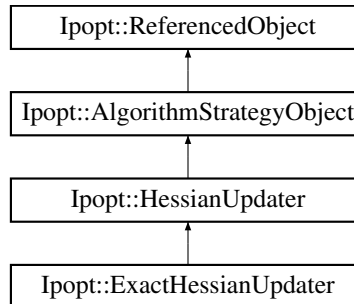
- [Algorithm/IpEquilibrationScaling.hpp](#)

6.37 Ipopt::ExactHessianUpdater Class Reference

Implementation of the [HessianUpdater](#) for the use of exact second derivatives.

```
#include <IpExactHessianUpdater.hpp>
```

Inheritance diagram for Ipopt::ExactHessianUpdater:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual void [UpdateHessian](#) ()
Update the Hessian based on the current information in IpData.

Constructors/Destructors

- [ExactHessianUpdater](#) ()
Default Constructor.
- virtual [~ExactHessianUpdater](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ExactHessianUpdater](#) (const [ExactHessianUpdater](#) &)
Copy Constructor.
- void [operator=](#) (const [ExactHessianUpdater](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.37.1 Detailed Description

Implementation of the [HessianUpdater](#) for the use of exact second derivatives.

Definition at line 20 of file IpExactHessianUpdater.hpp.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 Ipopt::ExactHessianUpdater::ExactHessianUpdater () [inline]

Default Constructor.

Definition at line 26 of file IpExactHessianUpdater.hpp.

6.37.2.2 virtual Ipopt::ExactHessianUpdater::~ExactHessianUpdater () [inline],[virtual]

Default destructor.

Definition at line 30 of file IpExactHessianUpdater.hpp.

6.37.2.3 Ipopt::ExactHessianUpdater::ExactHessianUpdater (const ExactHessianUpdater &) [private]

Copy Constructor.

6.37.3 Member Function Documentation

6.37.3.1 virtual bool Ipopt::ExactHessianUpdater::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::HessianUpdater](#).

6.37.3.2 virtual void Ipopt::ExactHessianUpdater::UpdateHessian () [virtual]

Update the Hessian based on the current information in IpData.

Implements [Ipopt::HessianUpdater](#).

6.37.3.3 void Ipopt::ExactHessianUpdater::operator= (const ExactHessianUpdater &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

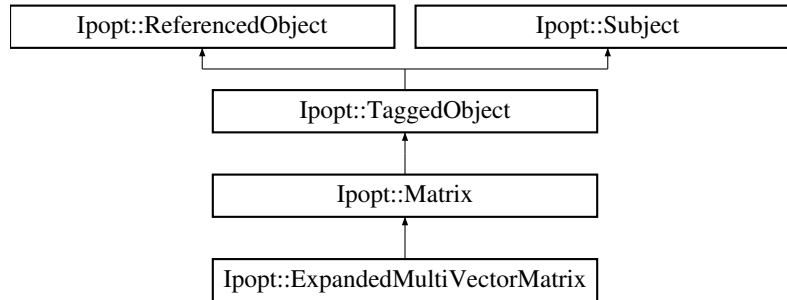
- [Algorithm/IpExactHessianUpdater.hpp](#)

6.38 Ipopt::ExpandedMultiVectorMatrix Class Reference

Class for Matrices with few rows that consists of Vectors, together with a premultiplied Expansion matrix.

```
#include <IpExpandedMultiVectorMatrix.hpp>
```

Inheritance diagram for Ipopt::ExpandedMultiVectorMatrix:



Public Member Functions

- **SmartPtr**
`< ExpandedMultiVectorMatrix > MakeNewExpandedMultiVectorMatrix () const`
- `void SetVector (Index i, SmartPtr< const Vector > vec)`
*Set a particular **Vector** at a given row position, replacing another vector if there has been one.*
- `SmartPtr< const Vector > GetVector (Index i) const`
*Get a **Vector** in a particular row as a const **Vector**.*
- `SmartPtr< const VectorSpace > RowVectorSpace () const`
***Vector** space for the rows.*
- `SmartPtr< const ExpandedMultiVectorMatrixSpace > ExpandedMultiVectorMatrixOwnerSpace () const`
*Return the **ExpandedMultiVectorMatrixSpace**.*
- `SmartPtr< const ExpansionMatrix > GetExpansionMatrix () const`
*Return the **Expansion matrix**.*

Constructors / Destructors

- `ExpandedMultiVectorMatrix (const ExpandedMultiVectorMatrixSpace *owner_space)`
Constructor, taking the owner_space.
- `virtual ~ExpandedMultiVectorMatrix ()`
Destructor.

Protected Member Functions

Overloaded methods from Matrix base class

- `virtual void MultVectorImpl (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix-vector multiply.
- `virtual void TransMultVectorImpl (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix(transpose) vector multiply.
- `virtual bool HasValidNumbersImpl () const`
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- `virtual void ComputeRowAMaxImpl (Vector &rows_norms, bool init) const`
Compute the max-norm of the rows in the matrix.
- `virtual void ComputeColAMaxImpl (Vector &cols_norms, bool init) const`
Compute the max-norm of the columns in the matrix.
- `virtual void PrintImpl (const Journalist &jnlst, EJournalLevel level, EJournalCategory category, const std::string &name, Index indent, const std::string &prefix) const`
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ExpandedMultiVectorMatrix](#) ()
Default Constructor.
- [ExpandedMultiVectorMatrix](#) (const [ExpandedMultiVectorMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [ExpandedMultiVectorMatrix](#) &)
Overloaded Equals Operator.

Private Attributes

- const
[ExpandedMultiVectorMatrixSpace](#) * [owner_space_](#)
- std::vector< [SmartPtr](#)< const
[Vector](#) > > [vecs_](#)
space for storing the const [Vector](#)'s

Additional Inherited Members

6.38.1 Detailed Description

Class for Matrices with few rows that consists of Vectors, together with a premultiplied Expansion matrix.

So, the matrix is $V^T P^T$. If P is NULL, it is assumed to be the identity matrix. If a row vector of V is NULL, it is assumed to be all zero. This is used to construct the KKT system with low-rank Hessian approximation.

Definition at line 29 of file `IpExpandedMultiVectorMatrix.hpp`.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 `Ipopt::ExpandedMultiVectorMatrix::ExpandedMultiVectorMatrix (const ExpandedMultiVectorMatrixSpace * owner_space)`

Constructor, taking the owner_space.

6.38.2.2 `virtual Ipopt::ExpandedMultiVectorMatrix::~~ExpandedMultiVectorMatrix () [inline],[virtual]`

Destructor.

Definition at line 41 of file `IpExpandedMultiVectorMatrix.hpp`.

6.38.2.3 `Ipopt::ExpandedMultiVectorMatrix::ExpandedMultiVectorMatrix () [private]`

Default Constructor.

6.38.2.4 `Ipopt::ExpandedMultiVectorMatrix::ExpandedMultiVectorMatrix (const ExpandedMultiVectorMatrix &) [private]`

Copy Constructor.

6.38.3 Member Function Documentation

6.38.3.1 `SmartPointer< ExpandedMultiVectorMatrix > Ipopt::ExpandedMultiVectorMatrix::MakeNewExpandedMultiVectorMatrix () const [inline]`

Definition at line 170 of file IpExpandedMultiVectorMatrix.hpp.

6.38.3.2 `void Ipopt::ExpandedMultiVectorMatrix::SetVector (Index i, SmartPtr< const Vector > vec)`

Set a particular [Vector](#) at a given row position, replacing another vector if there has been one.

6.38.3.3 `SmartPointer<const Vector> Ipopt::ExpandedMultiVectorMatrix::GetVector (Index i) const [inline]`

Get a [Vector](#) in a particular row as a const [Vector](#).

Definition at line 52 of file IpExpandedMultiVectorMatrix.hpp.

6.38.3.4 `SmartPointer< const VectorSpace > Ipopt::ExpandedMultiVectorMatrix::RowVectorSpace () const [inline]`

[Vector](#) space for the rows.

Definition at line 176 of file IpExpandedMultiVectorMatrix.hpp.

6.38.3.5 `SmartPointer< const ExpandedMultiVectorMatrixSpace > Ipopt::ExpandedMultiVectorMatrix::ExpandedMultiVectorMatrixOwnerSpace () const [inline]`

Return the [ExpandedMultiVectorMatrixSpace](#).

Definition at line 189 of file IpExpandedMultiVectorMatrix.hpp.

6.38.3.6 `SmartPointer< const ExpansionMatrix > Ipopt::ExpandedMultiVectorMatrix::GetExpansionMatrix () const [inline]`

Return the Expansion matrix.

If NULL, there is no expansion, the vector is used as is.

Definition at line 182 of file IpExpandedMultiVectorMatrix.hpp.

6.38.3.7 `virtual void Ipopt::ExpandedMultiVectorMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.38.3.8 `virtual void Ipopt::ExpandedMultiVectorMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.38.3.9 `virtual bool Ipopt::ExpandedMultiVectorMatrix::IsValidNumbersImpl () const [protected], [virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.38.3.10 `virtual void Ipopt::ExpandedMultiVectorMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const`
`[protected], [virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.38.3.11 `virtual void Ipopt::ExpandedMultiVectorMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const`
`[protected], [virtual]`

Compute the max-norm of the columns in the matrix.

The result is stored in cols_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.38.3.12 `virtual void Ipopt::ExpandedMultiVectorMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const`
`[protected], [virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.38.3.13 `void Ipopt::ExpandedMultiVectorMatrix::operator= (const ExpandedMultiVectorMatrix &)` `[private]`

Overloaded Equals Operator.

6.38.4 Member Data Documentation

6.38.4.1 `const ExpandedMultiVectorMatrixSpace* Ipopt::ExpandedMultiVectorMatrix::owner_space_` `[private]`

Definition at line 114 of file `IpExpandedMultiVectorMatrix.hpp`.

6.38.4.2 `std::vector<SmartPtr<const Vector> > Ipopt::ExpandedMultiVectorMatrix::vecs_` `[private]`

space for storing the const [Vector](#)'s

Definition at line 117 of file `IpExpandedMultiVectorMatrix.hpp`.

The documentation for this class was generated from the following file:

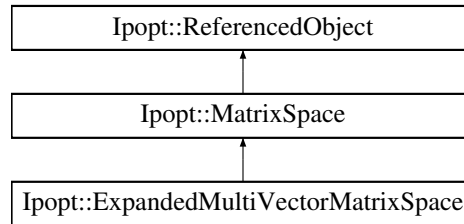
- [LinAlg/IpExpandedMultiVectorMatrix.hpp](#)

6.39 Ipopt::ExpandedMultiVectorMatrixSpace Class Reference

This is the matrix space for [ExpandedMultiVectorMatrix](#).

```
#include <IpExpandedMultiVectorMatrix.hpp>
```

Inheritance diagram for `Ipopt::ExpandedMultiVectorMatrixSpace`:



Public Member Functions

- `ExpandedMultiVectorMatrix * MakeNewExpandedMultiVectorMatrix () const`
Method for creating a new matrix of this specific type.
- `virtual Matrix * MakeNew () const`
Overloaded MakeNew method for the `MatrixSpace` base class.
- `SmartPointer< const VectorSpace > RowVectorSpace () const`
Accessor method for the `VectorSpace` for the rows.
- `SmartPointer< const ExpansionMatrix > GetExpansionMatrix () const`

Constructors / Destructors

- `ExpandedMultiVectorMatrixSpace (Index n_rows, const VectorSpace &vec_space, SmartPtr< const ExpansionMatrix > exp_matrix)`
Constructor, given the number of rows (i.e., Vectors to be stored) and given the `VectorSpace` for the Vectors.
- `virtual ~ExpandedMultiVectorMatrixSpace ()`
Destructor.

Private Attributes

- `SmartPointer< const VectorSpace > vec_space_`
- `SmartPointer< const ExpansionMatrix > exp_matrix_`

6.39.1 Detailed Description

This is the matrix space for `ExpandedMultiVectorMatrix`.

Definition at line 123 of file `IpExpandedMultiVectorMatrix.hpp`.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 `Ipopt::ExpandedMultiVectorMatrixSpace::ExpandedMultiVectorMatrixSpace (Index n_rows, const VectorSpace &vec_space, SmartPtr< const ExpansionMatrix > exp_matrix)`

Constructor, given the number of rows (i.e., Vectors to be stored) and given the `VectorSpace` for the Vectors.

6.39.2.2 `virtual Ipopt::ExpandedMultiVectorMatrixSpace::~~ExpandedMultiVectorMatrixSpace () [inline], [virtual]`

Destructor.

Definition at line 135 of file `IpExpandedMultiVectorMatrix.hpp`.

6.39.3 Member Function Documentation

6.39.3.1 `ExpandedMultiVectorMatrix* Ipopt::ExpandedMultiVectorMatrixSpace::MakeNewExpandedMultiVectorMatrix () const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 140 of file `IpExpandedMultiVectorMatrix.hpp`.

6.39.3.2 `virtual Matrix* Ipopt::ExpandedMultiVectorMatrixSpace::MakeNew () const [inline], [virtual]`

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 147 of file `IpExpandedMultiVectorMatrix.hpp`.

6.39.3.3 `SmartPointer<const VectorSpace> Ipopt::ExpandedMultiVectorMatrixSpace::RowVectorSpace () const [inline]`

Accessor method for the [VectorSpace](#) for the rows.

Definition at line 153 of file `IpExpandedMultiVectorMatrix.hpp`.

6.39.3.4 `SmartPointer<const ExpansionMatrix> Ipopt::ExpandedMultiVectorMatrixSpace::GetExpansionMatrix () const [inline]`

Definition at line 158 of file `IpExpandedMultiVectorMatrix.hpp`.

6.39.4 Member Data Documentation

6.39.4.1 `SmartPointer<const VectorSpace> Ipopt::ExpandedMultiVectorMatrixSpace::vec_space_ [private]`

Definition at line 164 of file `IpExpandedMultiVectorMatrix.hpp`.

6.39.4.2 `SmartPointer<const ExpansionMatrix> Ipopt::ExpandedMultiVectorMatrixSpace::exp_matrix_ [private]`

Definition at line 166 of file `IpExpandedMultiVectorMatrix.hpp`.

The documentation for this class was generated from the following file:

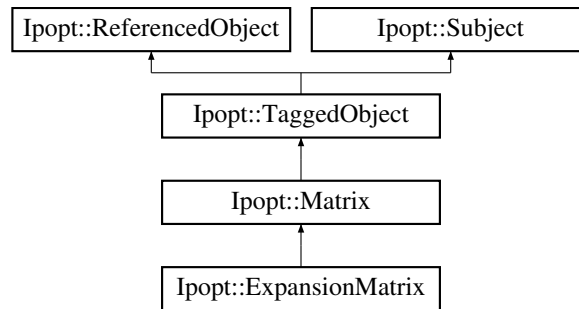
- [LinAlg/IpExpandedMultiVectorMatrix.hpp](#)

6.40 Ipopt::ExpansionMatrix Class Reference

Class for expansion/projection matrices.

```
#include <IpExpansionMatrix.hpp>
```

Inheritance diagram for `Ipopt::ExpansionMatrix`:



Public Member Functions

- `const Index * ExpandedPosIndices () const`
Return the vector of indices marking the expanded position.
- `const Index * CompressedPosIndices () const`
Return the vector of indices marking the compressed position.

Constructors / Destructors

- `ExpansionMatrix (const ExpansionMatrixSpace *owner_space)`
Constructor, taking the owner_space.
- `~ExpansionMatrix ()`
Destructor.

Protected Member Functions

- `void PrintImplOffset (const Journalist &jnlst, EJournalLevel level, EJournalCategory category, const std::string &name, Index indent, const std::string &prefix, Index row_offset, Index col_offset) const`

Overloaded methods from Matrix base class

- `virtual void MultVectorImpl (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix-vector multiply.
- `virtual void TransMultVectorImpl (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix(transpose) vector multiply.
- `virtual void AddMSinvZImpl (Number alpha, const Vector &S, const Vector &Z, Vector &X) const`
 $X = \beta X + \alpha * (\text{Matrix } S^{-1} \{ -1 \} Z).$
- `virtual void SinvBlrmZMTdBrlImpl (Number alpha, const Vector &S, const Vector &R, const Vector &Z, const Vector &D, Vector &X) const`
 $X = S^{-1} \{ -1 \} (r + \alpha * Z * M^T D).$
- `virtual void ComputeRowAMaxImpl (Vector &rows_norms, bool init) const`
Compute the max-norm of the rows in the matrix.
- `virtual void ComputeColAMaxImpl (Vector &cols_norms, bool init) const`
Compute the max-norm of the columns in the matrix.
- `virtual void PrintImpl (const Journalist &jnlst, EJournalLevel level, EJournalCategory category, const std::string &name, Index indent, const std::string &prefix) const`
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ExpansionMatrix](#) ()
Default Constructor.
- [ExpansionMatrix](#) (const [ExpansionMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [ExpansionMatrix](#) &)
Overloaded Equals Operator.

Private Attributes

- const [ExpansionMatrixSpace](#) * [owner_space_](#)

Friends

- class [ParExpansionMatrix](#)

Additional Inherited Members

6.40.1 Detailed Description

Class for expansion/projection matrices.

These matrices allow to lift a vector to a vector with larger dimension, keeping some elements of the larger vector zero. This operation is achieved by the MultVector operation. The transpose operation then filters some elements from a large vector into a smaller vector.

Definition at line 27 of file IpExpansionMatrix.hpp.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 Ipopt::ExpansionMatrix::ExpansionMatrix (const ExpansionMatrixSpace * owner_space)

Constructor, taking the owner_space.

6.40.2.2 Ipopt::ExpansionMatrix::~~ExpansionMatrix ()

Destructor.

6.40.2.3 Ipopt::ExpansionMatrix::ExpansionMatrix () [private]

Default Constructor.

6.40.2.4 Ipopt::ExpansionMatrix::ExpansionMatrix (const ExpansionMatrix &) [private]

Copy Constructor.

6.40.3 Member Function Documentation

6.40.3.1 `const Index * lpopt::ExpansionMatrix::ExpandedPosIndices () const` `[inline]`

Return the vector of indices marking the expanded position.

The result is the Index array (of length `NSmallVec=NCols()`) that stores the mapping from the small vector to the large vector. For each element $i=0,\dots,NSmallVec$ in the small vector, `ExpandedPosIndices()[i]` give the corresponding index in the large vector.

Definition at line 200 of file `lpExpansionMatrix.hpp`.

6.40.3.2 `const Index * lpopt::ExpansionMatrix::CompressedPosIndices () const` `[inline]`

Return the vector of indices marking the compressed position.

The result is the Index array (of length `NLargeVec=NRows()`) that stores the mapping from the large vector to the small vector. For each element $i=0,\dots,NLargeVec$ in the large vector, `CompressedPosIndices()[i]` gives the corresponding index in the small vector, unless `CompressedPosIndices()[i]` is negative.

Definition at line 206 of file `lpExpansionMatrix.hpp`.

6.40.3.3 `virtual void lpopt::ExpansionMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const` `[protected]`, `[virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements `lpopt::Matrix`.

6.40.3.4 `virtual void lpopt::ExpansionMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const` `[protected]`, `[virtual]`

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements `lpopt::Matrix`.

6.40.3.5 `virtual void lpopt::ExpansionMatrix::AddMSinvZImpl (Number alpha, const Vector & S, const Vector & Z, Vector & X) const` `[protected]`, `[virtual]`

$X = \beta * X + \alpha * (\text{Matrix} S^{-1} Z)$.

Specialized implementation.

Reimplemented from `lpopt::Matrix`.

6.40.3.6 `virtual void lpopt::ExpansionMatrix::SinvBlrmZMTdBrlImpl (Number alpha, const Vector & S, const Vector & R, const Vector & Z, const Vector & D, Vector & X) const` `[protected]`, `[virtual]`

$X = S^{-1} (r + \alpha * Z * M^T D)$.

Specialized implementation.

Reimplemented from `lpopt::Matrix`.

6.40.3.7 `virtual void lpopt::ExpansionMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` `[protected]`, `[virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in `rows_norms`. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

```
6.40.3.8 virtual void Ipopt::ExpansionMatrix::ComputeColAMaxImpl ( Vector & cols_norms, bool init ) const [protected],
[virtual]
```

Compute the max-norm of the columns in the matrix.

The result is stored in `cols_norms`. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

```
6.40.3.9 virtual void Ipopt::ExpansionMatrix::PrintImpl ( const Journalist & jnlst, EJournalLevel level, EJournalCategory
category, const std::string & name, Index indent, const std::string & prefix ) const [inline], [protected],
[virtual]
```

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

Definition at line 85 of file `IpExpansionMatrix.hpp`.

```
6.40.3.10 void Ipopt::ExpansionMatrix::PrintImplOffset ( const Journalist & jnlst, EJournalLevel level, EJournalCategory
category, const std::string & name, Index indent, const std::string & prefix, Index row_offset, Index col_offset )
const [protected]
```

```
6.40.3.11 void Ipopt::ExpansionMatrix::operator= ( const ExpansionMatrix & ) [private]
```

Overloaded Equals Operator.

6.40.4 Friends And Related Function Documentation

```
6.40.4.1 friend class ParExpansionMatrix [friend]
```

Definition at line 105 of file `IpExpansionMatrix.hpp`.

6.40.5 Member Data Documentation

```
6.40.5.1 const ExpansionMatrixSpace* Ipopt::ExpansionMatrix::owner_space_ [private]
```

Definition at line 126 of file `IpExpansionMatrix.hpp`.

The documentation for this class was generated from the following file:

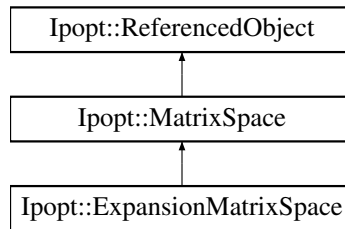
- [LinAlg/IpExpansionMatrix.hpp](#)

6.41 Ipopt::ExpansionMatrixSpace Class Reference

This is the matrix space for [ExpansionMatrix](#).

```
#include <IpExpansionMatrix.hpp>
```

Inheritance diagram for `Ipopt::ExpansionMatrixSpace`:



Public Member Functions

- [ExpansionMatrix](#) * [MakeNewExpansionMatrix](#) () const
Method for creating a new matrix of this specific type.
- virtual [Matrix](#) * [MakeNew](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.
- const [Index](#) * [ExpandedPosIndices](#) () const
Accessor Method to obtain the Index array (of length `NSmallVec=NCols()`) that stores the mapping from the small vector to the large vector.
- const [Index](#) * [CompressedPosIndices](#) () const
Accessor Method to obtain the Index array (of length `NLargeVec=NRows()`) that stores the mapping from the large vector to the small vector.

Constructors / Destructors

- [ExpansionMatrixSpace](#) ([Index](#) NLargeVec, [Index](#) NSmallVec, const [Index](#) *ExpPos, const int offset=0)
Constructor, given the list of elements of the large vector (of size NLargeVec) to be filtered into the small vector (of size NSmallVec).
- [~ExpansionMatrixSpace](#) ()
Destructor.

Private Attributes

- [Index](#) * [expanded_pos_](#)
- [Index](#) * [compressed_pos_](#)

6.41.1 Detailed Description

This is the matrix space for [ExpansionMatrix](#).

Definition at line 132 of file `IpExpansionMatrix.hpp`.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 `Ipopt::ExpansionMatrixSpace::ExpansionMatrixSpace (Index NLargeVec, Index NSmallVec, const Index * ExpPos, const int offset = 0)`

Constructor, given the list of elements of the large vector (of size NLargeVec) to be filtered into the small vector (of size NSmallVec).

For each $i=0..NSmallVec-1$ the i -th element of the small vector will be put into the `ExpPos[i]` position of the large vector. The position counting in the vector is assumed to start at 0 (C-like array notation).

6.41.2.2 Ipopt::ExpansionMatrixSpace::~~ExpansionMatrixSpace () [inline]

Destructor.

Definition at line 150 of file IpExpansionMatrix.hpp.

6.41.3 Member Function Documentation

6.41.3.1 ExpansionMatrix* Ipopt::ExpansionMatrixSpace::MakeNewExpansionMatrix () const [inline]

Method for creating a new matrix of this specific type.

Definition at line 158 of file IpExpansionMatrix.hpp.

6.41.3.2 virtual Matrix* Ipopt::ExpansionMatrixSpace::MakeNew () const [inline],[virtual]

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 165 of file IpExpansionMatrix.hpp.

6.41.3.3 const Index* Ipopt::ExpansionMatrixSpace::ExpandedPosIndices () const [inline]

Accessor Method to obtain the Index array (of length NSmallVec=[NCols\(\)](#)) that stores the mapping from the small vector to the large vector.

For each element $i=0,\dots,NSmallVec$ in the small vector, [ExpandedPosIndices\(\)\[i\]](#) give the corresponding index in the large vector.

Definition at line 176 of file IpExpansionMatrix.hpp.

6.41.3.4 const Index* Ipopt::ExpansionMatrixSpace::CompressedPosIndices () const [inline]

Accessor Method to obtain the Index array (of length NLargeVec=[NRows\(\)](#)) that stores the mapping from the large vector to the small vector.

For each element $i=0,\dots,NLargeVec$ in the large vector, [CompressedPosIndices\(\)\[i\]](#) gives the corresponding index in the small vector, unless [CompressedPosIndices\(\)\[i\]](#) is negative.

Definition at line 188 of file IpExpansionMatrix.hpp.

6.41.4 Member Data Documentation

6.41.4.1 Index* Ipopt::ExpansionMatrixSpace::expanded_pos_ [private]

Definition at line 194 of file IpExpansionMatrix.hpp.

6.41.4.2 Index* Ipopt::ExpansionMatrixSpace::compressed_pos_ [private]

Definition at line 195 of file IpExpansionMatrix.hpp.

The documentation for this class was generated from the following file:

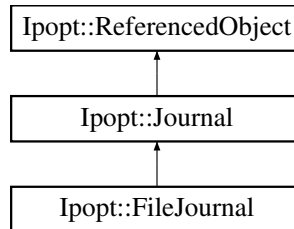
- [LinAlg/IpExpansionMatrix.hpp](#)

6.42 Ipopt::FileJournal Class Reference

[FileJournal](#) class.

```
#include <IpJournalist.hpp>
```

Inheritance diagram for Ipopt::FileJournal:



Public Member Functions

- [FileJournal](#) (const std::string &name, [EJournalLevel](#) default_level)
Constructor.
- virtual [~FileJournal](#) ()
Destructor.
- virtual bool [Open](#) (const char *fname)
Open a new file for the output location.

Protected Member Functions

Implementation version of Print methods - Overloaded from

[Journal](#) base class.

- virtual void [PrintImpl](#) ([EJournalCategory](#) category, [EJournalLevel](#) level, const char *str)
Print to the designated output location.
- virtual void [PrintfImpl](#) ([EJournalCategory](#) category, [EJournalLevel](#) level, const char *pformat, va_list ap)
Printf to the designated output location.
- virtual void [FlushBufferImpl](#) ()
Flush output buffer.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [FileJournal](#) ()
Default Constructor.
- [FileJournal](#) (const [FileJournal](#) &)
Copy Constructor.
- void [operator=](#) (const [FileJournal](#) &)
Overloaded Equals Operator.

Private Attributes

- FILE * [file_](#)

FILE pointer for the output destination.

6.42.1 Detailed Description

[FileJournal](#) class.

This is a particular [Journal](#) implementation that writes to a file for output. It can write to (stdout, stderr, or disk) by using "stdout" and "stderr" as filenames.

Definition at line 379 of file IpJournalist.hpp.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 Ipopt::FileJournal::FileJournal (const std::string & name, EJournalLevel default_level)

Constructor.

6.42.2.2 virtual Ipopt::FileJournal::~FileJournal () [virtual]

Destructor.

6.42.2.3 Ipopt::FileJournal::FileJournal () [private]

Default Constructor.

6.42.2.4 Ipopt::FileJournal::FileJournal (const FileJournal &) [private]

Copy Constructor.

6.42.3 Member Function Documentation

6.42.3.1 virtual bool Ipopt::FileJournal::Open (const char * fname) [virtual]

Open a new file for the output location.

Special Names: stdout means stdout, : stderr means stderr.

Return code is false only if the file with the given name could not be opened.

6.42.3.2 virtual void Ipopt::FileJournal::PrintImpl (EJournalCategory category, EJournalLevel level, const char * str) [protected], [virtual]

Print to the designated output location.

Implements [Ipopt::Journal](#).

6.42.3.3 virtual void Ipopt::FileJournal::PrintfImpl (EJournalCategory category, EJournalLevel level, const char * pformat, va_list ap) [protected], [virtual]

Printf to the designated output location.

Implements [Ipopt::Journal](#).

6.42.3.4 `virtual void Ipopt::FileJournal::FlushBufferImpl () [protected],[virtual]`

Flush output buffer.

Implements [Ipopt::Journal](#).

6.42.3.5 `void Ipopt::FileJournal::operator= (const FileJournal &) [private]`

Overloaded Equals Operator.

6.42.4 Member Data Documentation

6.42.4.1 `FILE* Ipopt::FileJournal::file_ [private]`

FILE pointer for the output destination.

Definition at line 434 of file `IpJournalist.hpp`.

The documentation for this class was generated from the following file:

- [Common/IpJournalist.hpp](#)

6.43 Ipopt::Filter Class Reference

Class for the filter.

```
#include <IpFilter.hpp>
```

Public Member Functions

- `bool Acceptable (std::vector< Number > vals) const`
Check acceptability of given coordinates with respect to the filter.
- `void AddEntry (std::vector< Number > vals, Index iteration)`
Add filter entry for given coordinates.
- `void Clear ()`
Delete all filter entries.
- `void Print (const Journalist &jnlst)`
Print current filter entries.

Constructors/Destructors

- `Filter (Index dim)`
Default Constructor.
- `~Filter ()`
Default Destructor.

Wrappers for 2-dimensional filter.

- `bool Acceptable (Number val1, Number val2) const`
- `void AddEntry (Number val1, Number val2, Index iteration)`

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `Filter ()`
Default Constructor.
- `Filter (const Filter &)`
Copy Constructor.
- `void operator= (const Filter &)`
Overloaded Equals Operator.

Private Attributes

- `Index dim_`
Dimension of the filter (number of coordinates per entry)
- `std::list< FilterEntry * > filter_list_`
List storing the filter entries.

6.43.1 Detailed Description

Class for the filter.

This class contains all filter entries. The entries are stored as the corner point, including the margin.

Definition at line 111 of file IpFilter.hpp.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 Ipopt::Filter::Filter (Index dim)

Default Constructor.

6.43.2.2 Ipopt::Filter::~Filter () [inline]

Default Destructor.

Definition at line 119 of file IpFilter.hpp.

6.43.2.3 Ipopt::Filter::Filter () [private]

Default Constructor.

6.43.2.4 Ipopt::Filter::Filter (const Filter &) [private]

Copy Constructor.

6.43.3 Member Function Documentation

6.43.3.1 bool Ipopt::Filter::Acceptable (std::vector< Number > vals) const

Check acceptability of given coordinates with respect to the filter.

Returns true, if pair is acceptable

6.43.3.2 `void Ipopt::Filter::AddEntry (std::vector< Number > vals, Index iteration)`

Add filter entry for given coordinates.

This will also delete all dominated entries in the current filter.

6.43.3.3 `bool Ipopt::Filter::Acceptable (Number val1, Number val2) const [inline]`

Definition at line 137 of file IpFilter.hpp.

6.43.3.4 `void Ipopt::Filter::AddEntry (Number val1, Number val2, Index iteration) [inline]`

Definition at line 146 of file IpFilter.hpp.

6.43.3.5 `void Ipopt::Filter::Clear ()`

Delete all filter entries.

6.43.3.6 `void Ipopt::Filter::Print (const Journalist & jnlst)`

Print current filter entries.

6.43.3.7 `void Ipopt::Filter::operator= (const Filter &) [private]`

Overloaded Equals Operator.

6.43.4 Member Data Documentation

6.43.4.1 `Index Ipopt::Filter::dim_ [private]`

Dimension of the filter (number of coordinates per entry)

Definition at line 181 of file IpFilter.hpp.

6.43.4.2 `std::list<FilterEntry*> Ipopt::Filter::filter_list_ [mutable],[private]`

List storing the filter entries.

Definition at line 184 of file IpFilter.hpp.

The documentation for this class was generated from the following file:

- Algorithm/[IpFilter.hpp](#)

6.44 Ipopt::FilterEntry Class Reference

Class for one filter entry.

```
#include <IpFilter.hpp>
```

Public Member Functions

- `bool Acceptable (std::vector< Number > vals) const`
Check acceptability of pair (*phi, theta*) with respect to this filter entry.

- bool **Dominated** (std::vector< **Number** > vals) const
Check if this entry is dominated by given coordinates.

Constructors/Destructors

- **FilterEntry** (std::vector< **Number** > vals, **Index** iter)
Constructor with the two components and the current iteration count.
- **~FilterEntry** ()
Default Destructor.

Accessor functions

- **Number** val (**Index** i) const
- **Index** iter () const

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- **FilterEntry** ()
Default Constructor.
- **FilterEntry** (const **FilterEntry** &)
Copy Constructor.
- void **operator=** (const **FilterEntry** &)
Overloaded Equals Operator.

Private Attributes

- std::vector< **Number** > **vals_**
values defining the coordinates of the entry
- const **Index** **iter_**
iteration number in which this entry was added to filter

6.44.1 Detailed Description

Class for one filter entry.

Definition at line 21 of file IpFilter.hpp.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 Ipopt::FilterEntry::FilterEntry (std::vector< **Number** > vals, **Index** iter)

Constructor with the two components and the current iteration count.

6.44.2.2 Ipopt::FilterEntry::~~FilterEntry ()

Default Destructor.

6.44.2.3 `Ipopt::FilterEntry::FilterEntry ()` [private]

Default Constructor.

6.44.2.4 `Ipopt::FilterEntry::FilterEntry (const FilterEntry &)` [private]

Copy Constructor.

6.44.3 Member Function Documentation

6.44.3.1 `bool Ipopt::FilterEntry::Acceptable (std::vector< Number > vals) const` [inline]

Check acceptability of pair (phi,theta) with respect to this filter entry.

Returns true, if pair is acceptable.

Definition at line 36 of file `IpFilter.hpp`.

6.44.3.2 `bool Ipopt::FilterEntry::Dominated (std::vector< Number > vals) const` [inline]

Check if this entry is dominated by given coordinates.

Returns true, if this entry is dominated.

Definition at line 56 of file `IpFilter.hpp`.

6.44.3.3 `Number Ipopt::FilterEntry::val (Index i) const` [inline]

Definition at line 74 of file `IpFilter.hpp`.

6.44.3.4 `Index Ipopt::FilterEntry::iter () const` [inline]

Definition at line 78 of file `IpFilter.hpp`.

6.44.3.5 `void Ipopt::FilterEntry::operator= (const FilterEntry &)` [private]

Overloaded Equals Operator.

6.44.4 Member Data Documentation

6.44.4.1 `std::vector<Number> Ipopt::FilterEntry::vals_` [private]

values defining the coordinates of the entry

Definition at line 103 of file `IpFilter.hpp`.

6.44.4.2 `const Index Ipopt::FilterEntry::iter_` [private]

iteration number in which this entry was added to filter

Definition at line 105 of file `IpFilter.hpp`.

The documentation for this class was generated from the following file:

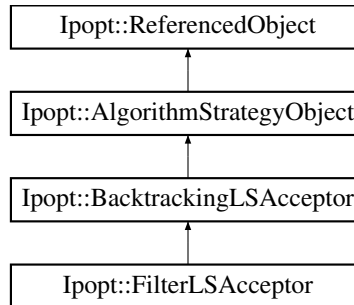
- [Algorithm/IpFilter.hpp](#)

6.45 Ipopt::FilterLSAcceptor Class Reference

[Filter](#) line search.

```
#include <IpFilterLSAcceptor.hpp>
```

Inheritance diagram for Ipopt::FilterLSAcceptor:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
InitializeImpl - overloaded from [AlgorithmStrategyObject](#).
- virtual void [Reset](#) ()
Reset the acceptor.
- virtual void [InitThisLineSearch](#) (bool in_watchdog)
Initialization for the next line search.
- virtual void [PrepareRestoPhaseStart](#) ()
Method that is called before the restoration phase is called.
- virtual [Number](#) [CalculateAlphaMin](#) ()
Method returning the lower bound on the trial step sizes.
- virtual bool [CheckAcceptabilityOfTrialPoint](#) ([Number](#) alpha_primal)
Method for checking if current trial point is acceptable.
- virtual bool [TrySecondOrderCorrection](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try a second order correction for the constraints.
- virtual bool [TryCorrector](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try higher order corrector (for fast local convergence).
- virtual char [UpdateForNextIteration](#) ([Number](#) alpha_primal_test)
Method for ending the current line search.
- virtual void [StartWatchDog](#) ()
Method for setting internal data if the watchdog procedure is started.
- virtual void [StopWatchDog](#) ()
Method for setting internal data if the watchdog procedure is stopped.

Constructors/Destructors

- [FilterLSAcceptor](#) (const [SmartPtr](#)< [PDSolver](#) > &pd_solver)
Constructor.
- virtual [~FilterLSAcceptor](#) ()

Default destructor.

Trial Point Accepting Methods. Used internally to check certain

acceptability criteria and used externally (by the restoration phase convergence check object, for instance)

- bool [IsAcceptableToCurrentIterate](#) (Number trial_barr, Number trial_theta, bool called_from_restoration=false) const
Checks if a trial point is acceptable to the current iterate.
- bool [IsAcceptableToCurrentFilter](#) (Number trial_barr, Number trial_theta) const
Checks if a trial point is acceptable to the current filter.

Static Public Member Functions

- static void [RegisterOptions](#) (SmartPtr< [RegisteredOptions](#) > roptions)
Methods for [OptionsList](#).

Private Member Functions

- bool [IsFtype](#) (Number alpha_primal_test)
Method for checking if the current step size satisfies the f-type switching condition.
- bool [ArmijoHolds](#) (Number alpha_primal_test)
Method for checking the Armijo condition, given a trial step size.
- void [AugmentFilter](#) ()
Augment the filter used on the current values of the barrier objective function and the constraint violation.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [FilterLSAcceptor](#) (const [FilterLSAcceptor](#) &)
Copy Constructor.
- void [operator=](#) (const [FilterLSAcceptor](#) &)
Overloaded Equals Operator.

Private Attributes

- [Filter filter_](#)
Filter with entries.

Filter information

- [Number theta_max_](#)
Upper bound on infeasibility.
- [Number theta_max_fact_](#)
- [Number theta_min_](#)
Infeasibility switching bound.
- [Number theta_min_fact_](#)

Information related to watchdog procedure

- [Number reference_theta_](#)
Constraint violation at the point with respect to which progress is to be made.
- [Number reference_barr_](#)
Barrier objective function at the point with respect to which progress is to be made.
- [Number reference_gradBarrTDelta_](#)
Barrier gradient transpose search direction at the point with respect to which progress is to be made.
- [Number watchdog_theta_](#)
Constraint violation at reference point.
- [Number watchdog_barr_](#)
Barrier objective function at reference point.
- [Number watchdog_gradBarrTDelta_](#)
Barrier gradient transpose search direction at reference point.

Filter reset stuff

- [Number last_rejection_due_to_filter_](#)
True, if last rejected was due to the filter.
- [Index count_successive_filter_rejections_](#)
Counter of successive iterations in which filter was reason for last rejection.
- [Index n_filter_resets_](#)
Counter for the filter resets done so far.

Strategy objective that are used

- [SmartPtr< PDSystemSolver > pd_solver_](#)

Parameters for the filter algorithm. Names as in the paper

- enum [CorrectorTypeEnum](#) { [NO_CORRECTOR](#) =0, [AFFINE_CORRECTOR](#), [PRIMAL_DUAL_CORRECTOR](#) }
enumeration for the corrector type
- [Number eta_phi_](#)
 η_ϕ
- [Number delta_](#)
 δ
- [Number s_phi_](#)
 s_ϕ
- [Number s_theta_](#)
 s_Θ
- [Number gamma_phi_](#)
 γ_ϕ
- [Number gamma_theta_](#)
 γ_Θ
- [Number alpha_min_frac_](#)
 γ_α
- [Index max_soc_](#)
Maximal number of second order correction steps.
- [Number kappa_soc_](#)
Required reduction in constraint violation before trying multiple second order correction steps κ_{soc} .
- [Number obj_max_inc_](#)
Maximal increase in objective function in orders of magnitude ($\log 10$).

- [CorrectorTypeEnum corrector_type_](#)
Type of corrector steps that should be tried.
- [Number corrector_compl_avg_red_fact_](#)
parameter in heuristic that determines whether corrector step should be tried.
- [bool skip_corr_if_neg_curv_](#)
Flag indicating whether the corrector should be skipped in an iteration in which negative curvature is detected.
- [bool skip_corr_in_monotone_mode_](#)
Flag indicating whether the corrector should be skipped during the monotone mu mode.
- [Index max_filter_resets_](#)
maximal allowed number of filter resets.
- [Index filter_reset_trigger_](#)
iteration counter trigger for filter reset.

Additional Inherited Members

6.45.1 Detailed Description

[Filter](#) line search.

This class implements the filter line search procedure.

Definition at line 23 of file `IpFilterLSAcceptor.hpp`.

6.45.2 Member Enumeration Documentation

6.45.2.1 `enum Ipopt::FilterLSAcceptor::CorrectorTypeEnum` `[private]`

enumeration for the corrector type

Enumerator

`NO_CORRECTOR`

`AFFINE_CORRECTOR`

`PRIMAL_DUAL_CORRECTOR`

Definition at line 199 of file `IpFilterLSAcceptor.hpp`.

6.45.3 Constructor & Destructor Documentation

6.45.3.1 `Ipopt::FilterLSAcceptor::FilterLSAcceptor (const SmartPtr< PDSystemSolver > &pd_solver)`

Constructor.

The [PDSystemSolver](#) object only needs to be provided (i.e. not NULL) if second order correction or corrector steps are to be used.

6.45.3.2 `virtual Ipopt::FilterLSAcceptor::~FilterLSAcceptor ()` `[virtual]`

Default destructor.

6.45.3.3 Ipopt::FilterLSAcceptor::FilterLSAcceptor (const FilterLSAcceptor &) [private]

Copy Constructor.

6.45.4 Member Function Documentation

6.45.4.1 virtual bool Ipopt::FilterLSAcceptor::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

InitializeImpl - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.2 virtual void Ipopt::FilterLSAcceptor::Reset () [virtual]

Reset the acceptor.

This function should be called if all previous information should be discarded when the line search is performed the next time. For example, this method should be called if the barrier parameter is changed.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.3 virtual void Ipopt::FilterLSAcceptor::InitThisLineSearch (bool in_watchdog) [virtual]

Initialization for the next line search.

The flag in_watchdog indicates if we are currently in an active watchdog procedure.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.4 virtual void Ipopt::FilterLSAcceptor::PrepareRestoPhaseStart () [virtual]

Method that is called before the restoration phase is called.

Here, we can set up things that are required in the termination test for the restoration phase, such as augmenting a filter.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.5 virtual Number Ipopt::FilterLSAcceptor::CalculateAlphaMin () [virtual]

Method returning the lower bound on the trial step sizes.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.6 virtual bool Ipopt::FilterLSAcceptor::CheckAcceptabilityOfTrialPoint (Number alpha_primal) [virtual]

Method for checking if current trial point is acceptable.

It is assumed that the delta information in ip_data is the search direction used in criteria. The primal trial point has to be set before the call.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.7 virtual bool Ipopt::FilterLSAcceptor::TrySecondOrderCorrection (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta) [virtual]

Try a second order correction for the constraints.

If the first trial step (with incoming alpha_primal) has been reject, this tries up to max_soc_ second order corrections for the constraints. Here, alpha_primal_test is the step size that has to be used in the filter acceptance tests. On output actual_delta_ has been set to the step including the second order correction if it has been accepted, otherwise it is

unchanged. If the SOC step has been accepted, `alpha_primal` has the fraction-to-the-boundary value for the SOC step on output. The return value is true, if a SOC step has been accepted.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.8 `virtual bool Ipopt::FilterLSAcceptor::TryCorrector (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta) [virtual]`

Try higher order corrector (for fast local convergence).

In contrast to a second order correction step, which tries to make an unacceptable point acceptable by improving constraint violation, this corrector step is tried even if the regular primal-dual step is acceptable.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.9 `virtual char Ipopt::FilterLSAcceptor::UpdateForNextIteration (Number alpha_primal_test) [virtual]`

Method for ending the current line search.

When it is called, the internal data should be updates, e.g., the filter might be augmented. `alpha_primal_test` is the value of alpha that has been used for in the acceptance test earlier.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.10 `virtual void Ipopt::FilterLSAcceptor::StartWatchDog () [virtual]`

Method for setting internal data if the watchdog procedure is started.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.11 `virtual void Ipopt::FilterLSAcceptor::StopWatchDog () [virtual]`

Method for setting internal data if the watchdog procedure is stopped.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.45.4.12 `bool Ipopt::FilterLSAcceptor::IsAcceptableToCurrentIterate (Number trial_barr, Number trial_theta, bool called_from_restoration = false) const`

Checks if a trial point is acceptable to the current iterate.

6.45.4.13 `bool Ipopt::FilterLSAcceptor::IsAcceptableToCurrentFilter (Number trial_barr, Number trial_theta) const`

Checks if a trial point is acceptable to the current filter.

6.45.4.14 `static void Ipopt::FilterLSAcceptor::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for [OptionsList](#).

6.45.4.15 `void Ipopt::FilterLSAcceptor::operator= (const FilterLSAcceptor &) [private]`

Overloaded Equals Operator.

6.45.4.16 `bool Ipopt::FilterLSAcceptor::IsFtype (Number alpha_primal_test) [private]`

Method for checking if the current step size satisfies the f-type switching condition.

Here, we use the search direction stored in `ip_data`

6.45.4.17 **bool** Ipopt::FilterLSAcceptor::ArmijoHolds (**Number** *alpha_primal_test*) [private]

Method for checking the Armijo condition, given a trial step size.

The test uses the search direction stored in ip_data, and the values of the functions at the trial point in ip_data.

6.45.4.18 **void** Ipopt::FilterLSAcceptor::AugmentFilter () [private]

Augment the filter used on the current values of the barrier objective function and the constraint violation.

6.45.5 Member Data Documentation

6.45.5.1 **Number** Ipopt::FilterLSAcceptor::theta_max_ [private]

Upper bound on infeasibility.

Definition at line 146 of file IpFilterLSAcceptor.hpp.

6.45.5.2 **Number** Ipopt::FilterLSAcceptor::theta_max_fact_ [private]

Definition at line 147 of file IpFilterLSAcceptor.hpp.

6.45.5.3 **Number** Ipopt::FilterLSAcceptor::theta_min_ [private]

Infeasibility switching bound.

Definition at line 150 of file IpFilterLSAcceptor.hpp.

6.45.5.4 **Number** Ipopt::FilterLSAcceptor::theta_min_fact_ [private]

Definition at line 151 of file IpFilterLSAcceptor.hpp.

6.45.5.5 **Number** Ipopt::FilterLSAcceptor::eta_phi_ [private]

η_ϕ

Definition at line 173 of file IpFilterLSAcceptor.hpp.

6.45.5.6 **Number** Ipopt::FilterLSAcceptor::delta_ [private]

δ

Definition at line 175 of file IpFilterLSAcceptor.hpp.

6.45.5.7 **Number** Ipopt::FilterLSAcceptor::s_phi_ [private]

s_ϕ

Definition at line 177 of file IpFilterLSAcceptor.hpp.

6.45.5.8 **Number** Ipopt::FilterLSAcceptor::s_theta_ [private]

s_θ

Definition at line 179 of file IpFilterLSAcceptor.hpp.

6.45.5.9 **Number** Ipopt::FilterLSAcceptor::gamma_phi_ [private]

γ_ϕ

Definition at line 181 of file IpFilterLSAcceptor.hpp.

6.45.5.10 `Number Ipopt::FilterLSAcceptor::gamma_theta_ [private]`

γ_θ

Definition at line 183 of file IpFilterLSAcceptor.hpp.

6.45.5.11 `Number Ipopt::FilterLSAcceptor::alpha_min_frac_ [private]`

γ_α

Definition at line 185 of file IpFilterLSAcceptor.hpp.

6.45.5.12 `Index Ipopt::FilterLSAcceptor::max_soc_ [private]`

Maximal number of second order correction steps.

Definition at line 187 of file IpFilterLSAcceptor.hpp.

6.45.5.13 `Number Ipopt::FilterLSAcceptor::kappa_soc_ [private]`

Required reduction in constraint violation before trying multiple second order correction steps κ_{soc} .

Definition at line 191 of file IpFilterLSAcceptor.hpp.

6.45.5.14 `Number Ipopt::FilterLSAcceptor::obj_max_inc_ [private]`

Maximal increase in objective function in orders of magnitude (log10).

If the log10(barrier objective function) is increased more than this compared to the current point, the trial point is rejected.

Definition at line 196 of file IpFilterLSAcceptor.hpp.

6.45.5.15 `CorrectorTypeEnum Ipopt::FilterLSAcceptor::corrector_type_ [private]`

Type of corrector steps that should be tried.

Definition at line 206 of file IpFilterLSAcceptor.hpp.

6.45.5.16 `Number Ipopt::FilterLSAcceptor::corrector_compl_avrg_red_fact_ [private]`

parameter in heuristic that determines whether corrector step

should be tried.

Definition at line 209 of file IpFilterLSAcceptor.hpp.

6.45.5.17 `bool Ipopt::FilterLSAcceptor::skip_corr_if_neg_curv_ [private]`

Flag indicating whether the corrector should be skipped in an iteration in which negative curvature is detected.

Definition at line 212 of file IpFilterLSAcceptor.hpp.

6.45.5.18 `bool Ipopt::FilterLSAcceptor::skip_corr_in_monotone_mode_ [private]`

Flag indicating whether the corrector should be skipped during the monotone mu mode.

Definition at line 215 of file IpFilterLSAcceptor.hpp.

6.45.5.19 Index Ipopt::FilterLSAcceptor::max_filter_resets_ [private]

maximal allowed number of filter resets.

Definition at line 217 of file IpFilterLSAcceptor.hpp.

6.45.5.20 Index Ipopt::FilterLSAcceptor::filter_reset_trigger_ [private]

iteration counter trigger for filter reset.

If the successive number of iterations in which the last rejected step was due to the filter, and max_filter_resets is non-zero, then the filter is reset.

Definition at line 222 of file IpFilterLSAcceptor.hpp.

6.45.5.21 Number Ipopt::FilterLSAcceptor::reference_theta_ [private]

Constraint violation at the point with respect to which progress is to be made.

Definition at line 229 of file IpFilterLSAcceptor.hpp.

6.45.5.22 Number Ipopt::FilterLSAcceptor::reference_barr_ [private]

Barrier objective function at the point with respect to which progress is to be made.

Definition at line 232 of file IpFilterLSAcceptor.hpp.

6.45.5.23 Number Ipopt::FilterLSAcceptor::reference_gradBarrTDelta_ [private]

Barrier gradient transpose search direction at the point with respect to which progress is to be made.

Definition at line 235 of file IpFilterLSAcceptor.hpp.

6.45.5.24 Number Ipopt::FilterLSAcceptor::watchdog_theta_ [private]

Constraint violation at reference point.

Definition at line 237 of file IpFilterLSAcceptor.hpp.

6.45.5.25 Number Ipopt::FilterLSAcceptor::watchdog_barr_ [private]

Barrier objective function at reference point.

Definition at line 239 of file IpFilterLSAcceptor.hpp.

6.45.5.26 Number Ipopt::FilterLSAcceptor::watchdog_gradBarrTDelta_ [private]

Barrier gradient transpose search direction at reference point.

Definition at line 241 of file IpFilterLSAcceptor.hpp.

6.45.5.27 Filter Ipopt::FilterLSAcceptor::filter_ [private]

[Filter](#) with entries.

Definition at line 245 of file IpFilterLSAcceptor.hpp.

6.45.5.28 Number Ipopt::FilterLSAcceptor::last_rejection_due_to_filter_ [private]

True, if last rejected was due to the filter.

Definition at line 250 of file IpFilterLSAcceptor.hpp.

6.45.5.29 Index Ipopt::FilterLSAcceptor::count_successive_filter_rejections_ [private]

Counter of successive iterations in which filter was reason for last rejection.

Definition at line 253 of file IpFilterLSAcceptor.hpp.

6.45.5.30 Index Ipopt::FilterLSAcceptor::n_filter_resets_ [private]

Counter for the filter resets done so far.

Definition at line 255 of file IpFilterLSAcceptor.hpp.

6.45.5.31 SmartPtr<PDSolver> Ipopt::FilterLSAcceptor::pd_solver_ [private]

Definition at line 260 of file IpFilterLSAcceptor.hpp.

The documentation for this class was generated from the following file:

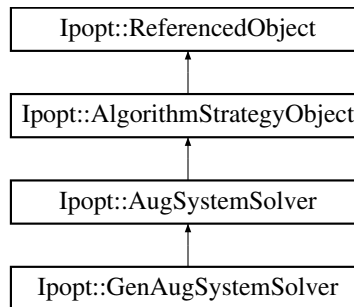
- [Algorithm/IpFilterLSAcceptor.hpp](#)

6.46 Ipopt::GenAugSystemSolver Class Reference

Solver for the augmented system using GenKKT solver interfaces.

```
#include <IpGenAugSystemSolver.hpp>
```

Inheritance diagram for Ipopt::GenAugSystemSolver:



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual [ESymSolverStatus](#) [MultiSolve](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) *J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) *J_d, const [Vector](#) *D_d, double delta_d, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_xV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_sV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_cV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_dV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_xV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_sV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_cV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_dV, bool check_NegEvals, [Index](#) numberOfNegEvals)
Set up the augmented system and solve it for a set of given right hand side - implementation for GenTMatrices and SymTMatrices.
- virtual [Index](#) [NumberOfNegEvals](#) () const
Number of negative eigenvalues detected during last solve.

- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.

Constructors/Destructors

- [GenAugSystemSolver](#) ([GenKKTSolverInterface](#) &[SolverInterface](#))
Constructor using only a linear solver object.
- virtual [~GenAugSystemSolver](#) ()
Default destructor.

Private Member Functions

- bool [AugmentedSystemChanged](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d)
Check the internal tags and decide if the passed variables are different from what is in the augmented_system_.
- void [UpdateTags](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d)

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [GenAugSystemSolver](#) ()
Default constructor.
- [GenAugSystemSolver](#) (const [GenAugSystemSolver](#) &)
Copy Constructor.
- void [operator=](#) (const [GenAugSystemSolver](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [GenKKTSolverInterface](#) > [solver_interface_](#)
The linear solver object that is to be used to solve the linear systems.

Tags and values to track in order to decide whether the

matrix has to be updated compared to the most recent call of the Set method.

- [TaggedObject::Tag](#) [w_tag_](#)
Tag for W matrix.
- double [w_factor_](#)
Most recent value of W_factor.
- [TaggedObject::Tag](#) [d_x_tag_](#)
Tag for D_x vector, representing the diagonal matrix D_x.
- double [delta_x_](#)
Most recent value of delta_x from Set method.
- [TaggedObject::Tag](#) [d_s_tag_](#)

- *Tag for D_s vector, representing the diagonal matrix D_s.*
- double `delta_s_`
Most recent value of delta_s from Set method.
- `TaggedObject::Tag j_c_tag_`
Tag for J_c matrix.
- `TaggedObject::Tag d_c_tag_`
Tag for D_c vector, representing the diagonal matrix D_c.
- double `delta_c_`
Most recent value of delta_c from Set method.
- `TaggedObject::Tag j_d_tag_`
Tag for J_d matrix.
- `TaggedObject::Tag d_d_tag_`
Tag for D_d vector, representing the diagonal matrix D_d.
- double `delta_d_`
Most recent value of delta_d from Set method.

Space for storing the diagonal matrices. If the matrix

hasn't changed, we can use it from the last call.

- `Number * dx_vals_copy_`
- `Number * ds_vals_copy_`
- `Number * dc_vals_copy_`
- `Number * dd_vals_copy_`

Algorithmic parameters

- bool `warm_start_same_structure_`
Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Additional Inherited Members

6.46.1 Detailed Description

Solver for the augmented system using `GenKKTsSolverInterfaces`.

This takes any [Vector](#) values out and provides `Number*`'s, but Matrices are provided as given from the [NLP](#).

Definition at line 22 of file `IpGenAugSystemSolver.hpp`.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 `Ipopt::GenAugSystemSolver::GenAugSystemSolver (GenKKTsSolverInterface & SolverInterface)`

Constructor using only a linear solver object.

6.46.2.2 `virtual Ipopt::GenAugSystemSolver::~~GenAugSystemSolver () [virtual]`

Default destructor.

6.46.2.3 `Ipopt::GenAugSystemSolver::GenAugSystemSolver () [private]`

Default constructor.

6.46.2.4 `Ipopt::GenAugSystemSolver::GenAugSystemSolver (const GenAugSystemSolver &) [private]`

Copy Constructor.

6.46.3 Member Function Documentation

6.46.3.1 `bool Ipopt::GenAugSystemSolver::InitializImpl (const OptionsList & options, const std::string & prefix)`
[virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AugSystemSolver](#).

6.46.3.2 `virtual ESymSolverStatus Ipopt::GenAugSystemSolver::MultiSolve (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix * J_c, const Vector * D_c, double delta_c, const Matrix * J_d, const Vector * D_d, double delta_d, std::vector< SmartPtr< const Vector > > & rhs_xV, std::vector< SmartPtr< const Vector > > & rhs_sV, std::vector< SmartPtr< const Vector > > & rhs_cV, std::vector< SmartPtr< const Vector > > & rhs_dV, std::vector< SmartPtr< Vector > > & sol_xV, std::vector< SmartPtr< Vector > > & sol_sV, std::vector< SmartPtr< Vector > > & sol_cV, std::vector< SmartPtr< Vector > > & sol_dV, bool check_NegEVals, Index numberOfNegEVals)` [virtual]

Set up the augmented system and solve it for a set of given right hand side - implementation for GenTMatrices and SymTMatrices.

Reimplemented from [Ipopt::AugSystemSolver](#).

6.46.3.3 `virtual Index Ipopt::GenAugSystemSolver::NumberOfNegEVals () const` [virtual]

Number of negative eigenvalues detected during last solve.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see ProvidesInertia).

Implements [Ipopt::AugSystemSolver](#).

6.46.3.4 `virtual bool Ipopt::GenAugSystemSolver::ProvidesInertia () const` [virtual]

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::AugSystemSolver](#).

6.46.3.5 `virtual bool Ipopt::GenAugSystemSolver::IncreaseQuality ()` [virtual]

Request to increase quality of solution for next solve.

Ask underlying linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::AugSystemSolver](#).

6.46.3.6 `void Ipopt::GenAugSystemSolver::operator= (const GenAugSystemSolver &)` [private]

Overloaded Equals Operator.

6.46.3.7 `bool Ipopt::GenAugSystemSolver::AugmentedSystemChanged (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix & J_c, const Vector * D_c, double delta_c, const Matrix & J_d, const Vector * D_d, double delta_d)` [private]

Check the internal tags and decide if the passed variables are different from what is in the augmented_system_.

6.46.3.8 `void Ipopt::GenAugSystemSolver::UpdateTags (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix & J_c, const Vector * D_c, double delta_c, const Matrix & J_d, const Vector * D_d, double delta_d)` [private]

6.46.4 Member Data Documentation

6.46.4.1 `SmartPtr<GenKKTsolverInterface> Ipopt::GenAugSystemSolver::solver_interface_` [private]

The linear solver object that is to be used to solve the linear systems.

Definition at line 136 of file `IpGenAugSystemSolver.hpp`.

6.46.4.2 `TaggedObject::Tag Ipopt::GenAugSystemSolver::w_tag_` [private]

Tag for *W* matrix.

If *W* has been given to Set as NULL, then this tag is set to 0

Definition at line 146 of file `IpGenAugSystemSolver.hpp`.

6.46.4.3 `double Ipopt::GenAugSystemSolver::w_factor_` [private]

Most recent value of *W_factor*.

Definition at line 148 of file `IpGenAugSystemSolver.hpp`.

6.46.4.4 `TaggedObject::Tag Ipopt::GenAugSystemSolver::d_x_tag_` [private]

Tag for *D_x* vector, representing the diagonal matrix *D_x*.

If *D_x* has been given to Set as NULL, then this tag is set to 0

Definition at line 152 of file `IpGenAugSystemSolver.hpp`.

6.46.4.5 `double Ipopt::GenAugSystemSolver::delta_x_` [private]

Most recent value of *delta_x* from Set method.

Definition at line 154 of file `IpGenAugSystemSolver.hpp`.

6.46.4.6 `TaggedObject::Tag Ipopt::GenAugSystemSolver::d_s_tag_` [private]

Tag for *D_s* vector, representing the diagonal matrix *D_s*.

If *D_s* has been given to Set as NULL, then this tag is set to 0

Definition at line 158 of file `IpGenAugSystemSolver.hpp`.

6.46.4.7 `double Ipopt::GenAugSystemSolver::delta_s_` [private]

Most recent value of *delta_s* from Set method.

Definition at line 160 of file `IpGenAugSystemSolver.hpp`.

6.46.4.8 `TaggedObject::Tag Ipopt::GenAugSystemSolver::j_c_tag_` [private]

Tag for *J_c* matrix.

If *J_c* has been given to Set as NULL, then this tag is set to 0

Definition at line 164 of file `IpGenAugSystemSolver.hpp`.

6.46.4.9 TaggedObject::Tag Ipopt::GenAugSystemSolver::d_c_tag_ [private]

Tag for D_c vector, representing the diagonal matrix D_c.

If D_c has been given to Set as NULL, then this tag is set to 0

Definition at line 168 of file IpGenAugSystemSolver.hpp.

6.46.4.10 double Ipopt::GenAugSystemSolver::delta_c_ [private]

Most recent value of delta_c from Set method.

Definition at line 170 of file IpGenAugSystemSolver.hpp.

6.46.4.11 TaggedObject::Tag Ipopt::GenAugSystemSolver::j_d_tag_ [private]

Tag for J_d matrix.

If J_d has been given to Set as NULL, then this tag is set to 0

Definition at line 174 of file IpGenAugSystemSolver.hpp.

6.46.4.12 TaggedObject::Tag Ipopt::GenAugSystemSolver::d_d_tag_ [private]

Tag for D_d vector, representing the diagonal matrix D_d.

If D_d has been given to Set as NULL, then this tag is set to 0

Definition at line 178 of file IpGenAugSystemSolver.hpp.

6.46.4.13 double Ipopt::GenAugSystemSolver::delta_d_ [private]

Most recent value of delta_d from Set method.

Definition at line 180 of file IpGenAugSystemSolver.hpp.

6.46.4.14 Number* Ipopt::GenAugSystemSolver::dx_vals_copy_ [private]

Definition at line 186 of file IpGenAugSystemSolver.hpp.

6.46.4.15 Number* Ipopt::GenAugSystemSolver::ds_vals_copy_ [private]

Definition at line 187 of file IpGenAugSystemSolver.hpp.

6.46.4.16 Number* Ipopt::GenAugSystemSolver::dc_vals_copy_ [private]

Definition at line 188 of file IpGenAugSystemSolver.hpp.

6.46.4.17 Number* Ipopt::GenAugSystemSolver::dd_vals_copy_ [private]

Definition at line 189 of file IpGenAugSystemSolver.hpp.

6.46.4.18 bool Ipopt::GenAugSystemSolver::warm_start_same_structure_ [private]

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 196 of file IpGenAugSystemSolver.hpp.

The documentation for this class was generated from the following file:

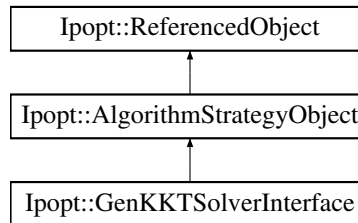
- [Algorithm/IpGenAugSystemSolver.hpp](#)

6.47 Ipopt::GenKKTSolverInterface Class Reference

Base class for interfaces to symmetric indefinite linear solvers for generic matrices.

```
#include <IpGenKKTSolverInterface.hpp>
```

Inheritance diagram for Ipopt::GenKKTSolverInterface:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)

Constructor/Destructor

- [GenKKTSolverInterface](#) ()
- virtual [~GenKKTSolverInterface](#) ()

Methods for requesting solution of the linear system.

- virtual [ESymSolverStatus MultiSolve](#) (bool new_matrix, [Index](#) n_x, [Index](#) n_c, [Index](#) n_d, [SmartPtr](#)< const [SymMatrix](#) > W, [SmartPtr](#)< const [Matrix](#) > Jac_c, [SmartPtr](#)< const [Matrix](#) > Jac_d, const [Number](#) *D_x, const [Number](#) *D_s, const [Number](#) *D_c, const [Number](#) *D_d, [Number](#) delta_x, [Number](#) delta_s, [Number](#) delta_c, [Number](#) delta_d, [Index](#) n_rhs, [Number](#) *rhssol, bool check_NegEVals, [Index](#) numberOfNegEVals)=0
Solve operation for multiple right hand sides.
- virtual [Index NumberOfNegEVals](#) () const =0
Number of negative eigenvalues detected during last factorization.
- virtual bool [IncreaseQuality](#) ()=0
Request to increase quality of solution for next solve.
- virtual bool [ProvidesInertia](#) () const =0
Query whether inertia is computed by linear solver.

Additional Inherited Members

6.47.1 Detailed Description

Base class for interfaces to symmetric indefinite linear solvers for generic matrices.

Definition at line 20 of file IpGenKKTSolverInterface.hpp.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 Ipopt::GenKKTSolverInterface::GenKKTSolverInterface () [inline]

Definition at line 25 of file IpGenKKTSolverInterface.hpp.

6.47.2.2 virtual Ipopt::GenKKTSolverInterface::~~GenKKTSolverInterface () [inline],[virtual]

Definition at line 28 of file IpGenKKTSolverInterface.hpp.

6.47.3 Member Function Documentation

6.47.3.1 virtual bool Ipopt::GenKKTSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix)
[pure virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

6.47.3.2 virtual ESymSolverStatus Ipopt::GenKKTSolverInterface::MultiSolve (bool new_matrix, Index n_x, Index n_c, Index n_d, SmartPtr< const SymMatrix > W, SmartPtr< const Matrix > Jac_c, SmartPtr< const Matrix > Jac_d, const Number * D_x, const Number * D_s, const Number * D_c, const Number * D_d, Number delta_x, Number delta_s, Number delta_c, Number delta_d, Index n_rhs, Number * rhssol, bool check_NegEVals, Index numberOfNegEVals) [pure virtual]

Solve operation for multiple right hand sides.

The linear system is of the form

$$\begin{bmatrix} W + D_x + \delta_x I & 0 & J_c^T & J_d^T \\ 0 & D_s + \delta_s I & 0 & -I \\ J_c & 0 & D_c - \delta_c I & 0 \\ J_d & -I & 0 & D_d - \delta_d I \end{bmatrix} \begin{pmatrix} sol_x \\ sol_s \\ sol_c \\ sol_d \end{pmatrix} = \begin{pmatrix} rhs_x \\ rhs_s \\ rhs_c \\ rhs_d \end{pmatrix}$$

(see also [AugSystemSolver](#)).

The return code is SYMSOLV_SUCCESS if the factorization and solves were successful, SYMSOLV_SINGULAR if the linear system is singular, and SYMSOLV_WRONG_INERTIA if check_NegEVals is true and the number of negative eigenvalues in the matrix does not match numberOfNegEVals. If SYMSOLV_CALL_AGAIN is returned, then the calling function will request the pointer for the array for storing a again (with GetValuesPtr), write the values of the nonzero elements into it, and call this MultiSolve method again with the same right-hand sides. (This can be done, for example, if the linear solver realized it does not have sufficient memory and needs to redo the factorization; e.g., for MA27.)

The number of right-hand sides is given by nrhs, the values of the right-hand sides are given in rhs_vals (one full right-hand side stored immediately after the other), and solutions are to be returned in the same array.

check_NegEVals will not be chosen true, if [ProvidesInertia\(\)](#) returns false.

Parameters

<i>new_matrix</i>	If this flag is false, the same matrix as in the most recent call is given to the solver again
<i>n_x</i>	Dimension of D_x
<i>n_c</i>	Dimension of D_s and D_c
<i>n_d</i>	Dimension of D_d
<i>W</i>	Hessian of Lagrangian (as given by NLP)
<i>Jac_c</i>	Jacobian of equality constraints (as given by NLP)
<i>Jac_d</i>	Jacobian of inequality constraints (as given by NLP)
<i>D_x</i>	Array with the elements D_x (if NULL, assume all zero)
<i>D_s</i>	Array with the elements D_s (if NULL, assume all zero)

<i>D_c</i>	Array with the elements <i>D_c</i> (if NULL, assume all zero)
<i>D_d</i>	Array with the elements <i>D_d</i> (if NULL, assume all zero)
<i>delta_x</i>	δ_x
<i>delta_s</i>	δ_s
<i>delta_c</i>	δ_c
<i>delta_d</i>	δ_d
<i>n_rhs</i>	Number of right hand sides
<i>rhssol</i>	On input, this contains the right hand sides, and on successful termination of the solver, the solutions are expected in there on return. At the moment, the order is x,d,c,s, but this can be made flexible and chosen according to an option.
<i>check_NegEVals</i>	if true, we want to ensure that the inertia is correct
<i>numberOfNegEVals</i>	Required number of negative eigenvalues if <i>check_NegEVals</i> is true

6.47.3.3 `virtual Index Ipopt::GenKKTSolverInterface::NumberOfNegEVals () const [pure virtual]`

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see *ProvidesInertia*).

6.47.3.4 `virtual bool Ipopt::GenKKTSolverInterface::IncreaseQuality () [pure virtual]`

Request to increase quality of solution for next solve.

The calling class asks linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

6.47.3.5 `virtual bool Ipopt::GenKKTSolverInterface::ProvidesInertia () const [pure virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

The documentation for this class was generated from the following file:

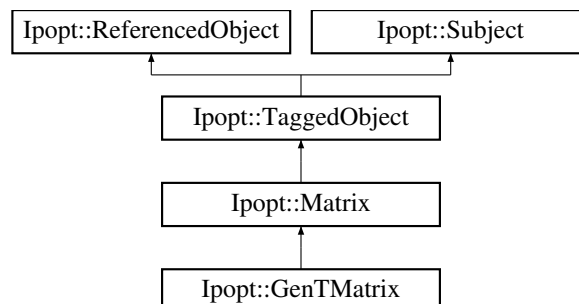
- [Algorithm/LinearSolvers/IpGenKKTSolverInterface.hpp](#)

6.48 Ipopt::GenTMatrix Class Reference

Class for general matrices stored in triplet format.

```
#include <IpGenTMatrix.hpp>
```

Inheritance diagram for Ipopt::GenTMatrix:



Public Member Functions

Constructors / Destructors

- [GenTMatrix](#) (const [GenTMatrixSpace](#) *owner_space)
Constructor, taking the owner_space.
- [~GenTMatrix](#) ()
Destructor.

Changing the Values.

- void [SetValues](#) (const [Number](#) *Values)
Set values of nonzero elements.

Accessor Methods

- [Index Nonzeros](#) () const
Number of nonzero entries.
- const [Index](#) * [Irows](#) () const
Array with Row indices (counting starts at 1)
- const [Index](#) * [Jcols](#) () const
Array with Column indices (counting starts at 1)
- const [Number](#) * [Values](#) () const
Array with nonzero values (const version).
- [Number](#) * [Values](#) ()
Array with the nonzero values of this matrix (non-const version).

Protected Member Functions

- void [PrintImplOffset](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix, [Index](#) offset) const

Overloaded methods from Matrix base class

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual void [TransMultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix(transpose) vector multiply.
- virtual bool [IsValidNumbersImpl](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &cols_norms, bool init) const
Compute the max-norm of the columns in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [GenTMatrix](#) ()
Default Constructor.
- [GenTMatrix](#) (const [GenTMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [GenTMatrix](#) &)
Overloaded Equals Operator.

Private Attributes

- const [GenTMatrixSpace](#) * [owner_space_](#)
Copy of the owner space as a [GenTMatrixSpace](#) instead of a [MatrixSpace](#).
- [Number](#) * [values_](#)
Values of nonzeros.
- bool [initialized_](#)
Flag for Initialization.

Friends

- class [ParGenMatrix](#)

Additional Inherited Members

6.48.1 Detailed Description

Class for general matrices stored in triplet format.

In the triplet format, the nonzeros elements of a general matrix is stored in three arrays, Irow, Jcol, and Values, all of length Nonzeros. The first two arrays indicate the location of a non-zero element (row and column indices), and the last array stores the value at that location. If nonzero elements are listed more than once, their values are added.

The structure of the nonzeros (i.e. the arrays Irow and Jcol) cannot be changed after the matrix can be initialized. Only the values of the nonzero elements can be modified.

Note that the first row and column of a matrix has index 1, not 0.

Definition at line 36 of file IpGenTMatrix.hpp.

6.48.2 Constructor & Destructor Documentation

6.48.2.1 Ipopt::GenTMatrix::GenTMatrix (const [GenTMatrixSpace](#) * [owner_space](#))

Constructor, taking the owner_space.

6.48.2.2 Ipopt::GenTMatrix::~~GenTMatrix ()

Destructor.

6.48.2.3 Ipopt::GenTMatrix::GenTMatrix () [private]

Default Constructor.

6.48.2.4 Ipopt::GenTMatrix::GenTMatrix (const GenTMatrix &) [private]

Copy Constructor.

6.48.3 Member Function Documentation

6.48.3.1 void Ipopt::GenTMatrix::SetValues (const Number * Values)

Set values of nonzero elements.

The values of the nonzero elements are copied from the incoming Number array. Important: It is assume that the order of the values in Values corresponds to the one of lrn and jcn given to one of the constructors above.

6.48.3.2 Index Ipopt::GenTMatrix::Nonzeros () const [inline]

Number of nonzero entries.

Definition at line 245 of file IpGenTMatrix.hpp.

6.48.3.3 const Index * Ipopt::GenTMatrix::lrows () const [inline]

Array with Row indices (counting starts at 1)

Definition at line 251 of file IpGenTMatrix.hpp.

6.48.3.4 const Index * Ipopt::GenTMatrix::jcols () const [inline]

Array with Column indices (counting starts at 1)

Definition at line 257 of file IpGenTMatrix.hpp.

6.48.3.5 const Number* Ipopt::GenTMatrix::Values () const [inline]

Array with nonzero values (const version).

Definition at line 73 of file IpGenTMatrix.hpp.

6.48.3.6 Number* Ipopt::GenTMatrix::Values () [inline]

Array with the nonzero values of this matrix (non-const version).

Use this method only if you are intending to change the values, because the [GenTMatrix](#) will be marked as changed.

Definition at line 82 of file IpGenTMatrix.hpp.

6.48.3.7 virtual void Ipopt::GenTMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.48.3.8 `virtual void Ipopt::GenTMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.48.3.9 `virtual bool Ipopt::GenTMatrix::IsValidNumbersImpl () const [protected], [virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.48.3.10 `virtual void Ipopt::GenTMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const [protected], [virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in *rows_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.48.3.11 `virtual void Ipopt::GenTMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const [protected], [virtual]`

Compute the max-norm of the columns in the matrix.

The result is stored in *cols_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.48.3.12 `virtual void Ipopt::GenTMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const [inline], [protected], [virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

Definition at line 107 of file IpGenTMatrix.hpp.

6.48.3.13 `void Ipopt::GenTMatrix::PrintImplOffset (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix, Index offset) const [protected]`

6.48.3.14 `void Ipopt::GenTMatrix::operator= (const GenTMatrix &) [private]`

Overloaded Equals Operator.

6.48.4 Friends And Related Function Documentation

6.48.4.1 `friend class ParGenMatrix [friend]`

Definition at line 126 of file IpGenTMatrix.hpp.

6.48.5 Member Data Documentation

6.48.5.1 `const GenTMatrixSpace* Ipopt::GenTMatrix::owner_space_ [private]`

Copy of the owner space as a [GenTMatrixSpace](#) instead of a [MatrixSpace](#).

Definition at line 150 of file `IpGenTMatrix.hpp`.

6.48.5.2 `Number* Ipopt::GenTMatrix::values_ [private]`

Values of nonzeros.

Definition at line 153 of file `IpGenTMatrix.hpp`.

6.48.5.3 `bool Ipopt::GenTMatrix::initialized_ [private]`

Flag for Initialization.

Definition at line 156 of file `IpGenTMatrix.hpp`.

The documentation for this class was generated from the following file:

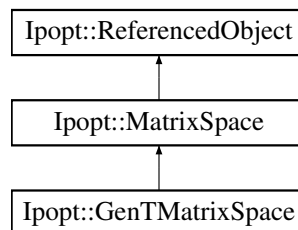
- `LinAlg/TMatrices/IpGenTMatrix.hpp`

6.49 Ipopt::GenTMatrixSpace Class Reference

This is the matrix space for a [GenTMatrix](#) with fixed sparsity structure.

```
#include <IpGenTMatrix.hpp>
```

Inheritance diagram for `Ipopt::GenTMatrixSpace`:



Public Member Functions

- [GenTMatrix](#) * [MakeNewGenTMatrix](#) () const
Method for creating a new matrix of this specific type.
- virtual [Matrix](#) * [MakeNew](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.

Constructors / Destructors

- [GenTMatrixSpace](#) ([Index](#) nRows, [Index](#) nCols, [Index](#) nonZeros, const [Index](#) *iRows, const [Index](#) *jCols)
Constructor, given the number of rows and columns, as well as the number of nonzeros and the position of the nonzero elements.
- [~GenTMatrixSpace](#) ()
Destructor.

Methods describing Matrix structure

- [Index Nonzeros](#) () const
Number of non-zeros in the sparse matrix.
- const [Index](#) * [Irows](#) () const
Row index of each non-zero element (counting starts at 1)
- const [Index](#) * [Jcols](#) () const
Column index of each non-zero element (counting starts at 1)

Private Member Functions

- [Number](#) * [AllocateInternalStorage](#) () const
This method is only for the [GenTMatrix](#) to call in order to allocate internal storage.
- void [FreeInternalStorage](#) ([Number](#) *values) const
This method is only for the [GenTMatrix](#) to call in order to de-allocate internal storage.

Private Attributes

Sparsity structure of matrices generated by this matrix

space.

- const [Index](#) [nonZeros_](#)
- [Index](#) * [jCols_](#)
- [Index](#) * [iRows_](#)

Friends

- class [GenTMatrix](#)

6.49.1 Detailed Description

This is the matrix space for a [GenTMatrix](#) with fixed sparsity structure.

The sparsity structure is stored here in the matrix space.

Definition at line 164 of file [IpGenTMatrix.hpp](#).

6.49.2 Constructor & Destructor Documentation

6.49.2.1 [Ipopt::GenTMatrixSpace::GenTMatrixSpace \(\[Index\]\(#\) *nRows*, \[Index\]\(#\) *nCols*, \[Index\]\(#\) *nonZeros*, const \[Index\]\(#\) * *iRows*, const \[Index\]\(#\) * *jCols* \)](#)

Constructor, given the number of rows and columns, as well as the number of nonzeros and the position of the nonzero elements.

Note that the counting of the nonzeros starts a 1, i.e., [iRows](#)[*i*]==1 and [jCols](#)[*i*]==1 refers to the first element in the first row. This is in accordance with the HSL data structure.

6.49.2.2 [Ipopt::GenTMatrixSpace::~~GenTMatrixSpace \(\)](#) [[inline](#)]

Destructor.

Definition at line 181 of file [IpGenTMatrix.hpp](#).

6.49.3 Member Function Documentation

6.49.3.1 GenTMatrix* Ipopt::GenTMatrixSpace::MakeNewGenTMatrix () const [inline]

Method for creating a new matrix of this specific type.

Definition at line 189 of file IpGenTMatrix.hpp.

6.49.3.2 virtual Matrix* Ipopt::GenTMatrixSpace::MakeNew () const [inline],[virtual]

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 196 of file IpGenTMatrix.hpp.

6.49.3.3 Index Ipopt::GenTMatrixSpace::Nonzeros () const [inline]

Number of non-zeros in the sparse matrix.

Definition at line 204 of file IpGenTMatrix.hpp.

6.49.3.4 const Index* Ipopt::GenTMatrixSpace::Irows () const [inline]

Row index of each non-zero element (counting starts at 1)

Definition at line 210 of file IpGenTMatrix.hpp.

6.49.3.5 const Index* Ipopt::GenTMatrixSpace::Jcols () const [inline]

Column index of each non-zero element (counting starts at 1)

Definition at line 216 of file IpGenTMatrix.hpp.

6.49.3.6 Number* Ipopt::GenTMatrixSpace::AllocateInternalStorage () const [private]

This method is only for the [GenTMatrix](#) to call in order to allocate internal storage.

6.49.3.7 void Ipopt::GenTMatrixSpace::FreeInternalStorage (Number * values) const [private]

This method is only for the [GenTMatrix](#) to call in order to de-allocate internal storage.

6.49.4 Friends And Related Function Documentation

6.49.4.1 friend class GenTMatrix [friend]

Definition at line 240 of file IpGenTMatrix.hpp.

6.49.5 Member Data Documentation

6.49.5.1 const Index Ipopt::GenTMatrixSpace::nonZeros_ [private]

Definition at line 227 of file IpGenTMatrix.hpp.

6.49.5.2 Index* Ipopt::GenTMatrixSpace::jCols_ [private]

Definition at line 228 of file IpGenTMatrix.hpp.

6.49.5.3 Index* Ipopt::GenTMatrixSpace::iRows_ [private]

Definition at line 229 of file IpGenTMatrix.hpp.

The documentation for this class was generated from the following file:

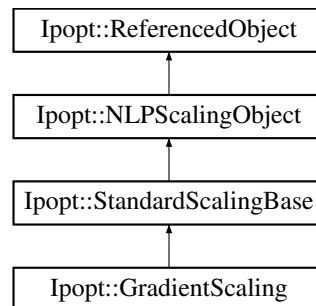
- LinAlg/TMatrices/IpGenTMatrix.hpp

6.50 Ipopt::GradientScaling Class Reference

This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point.

```
#include <IpGradientScaling.hpp>
```

Inheritance diagram for Ipopt::GradientScaling:



Public Member Functions

Constructors/Destructors

- [GradientScaling](#) (const [SmartPtr](#)< [NLP](#) > &nlp)
- virtual [~GradientScaling](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &options)
Methods for IpoptType.

Protected Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
Initialize the object from the options.
- virtual void [DetermineScalingParametersImpl](#) (const [SmartPtr](#)< const [VectorSpace](#) > x_space, const [SmartPtr](#)< const [VectorSpace](#) > c_space, const [SmartPtr](#)< const [VectorSpace](#) > d_space, const [SmartPtr](#)< const [MatrixSpace](#) > jac_c_space, const [SmartPtr](#)< const [MatrixSpace](#) > jac_d_space, const [SmartPtr](#)< const [SymMatrixSpace](#) > h_space, const [Matrix](#) &Px_L, const [Vector](#) &x_L, const [Matrix](#) &Px_U, const [Vector](#) &x_U, [Number](#) &df, [SmartPtr](#)< [Vector](#) > &dx, [SmartPtr](#)< [Vector](#) > &dc, [SmartPtr](#)< [Vector](#) > &dd)

This is the method that has to be overloaded by a particular scaling method that somehow computes the scaling vectors dx, dc, and dd.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [GradientScaling](#) (const [GradientScaling](#) &)
Copy Constructor.
- void [operator=](#) (const [GradientScaling](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr< NLP > nlp_](#)
pointer to the [NLP](#) to get scaling parameters
- [Number scaling_max_gradient_](#)
maximum allowed gradient before scaling is performed
- [Number scaling_obj_target_gradient_](#)
target size of norm for objective gradient
- [Number scaling_constr_target_gradient_](#)
target size of norm for constraint gradients
- [Number scaling_min_value_](#)
minimum value of a scaling parameter

6.50.1 Detailed Description

This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point.

Definition at line 21 of file IpGradientScaling.hpp.

6.50.2 Constructor & Destructor Documentation

6.50.2.1 Ipopt::GradientScaling::GradientScaling (const SmartPtr< NLP > & nlp) [inline]

Definition at line 26 of file IpGradientScaling.hpp.

6.50.2.2 virtual Ipopt::GradientScaling::~GradientScaling () [inline],[virtual]

Default destructor.

Definition at line 33 of file IpGradientScaling.hpp.

6.50.2.3 Ipopt::GradientScaling::GradientScaling (const GradientScaling &) [private]

Copy Constructor.

6.50.3 Member Function Documentation

6.50.3.1 `static void Ipopt::GradientScaling::RegisterOptions (const SmartPtr< RegisteredOptions > & roptions)`
[static]

Methods for IpoptType.

Register the options for this class

6.50.3.2 `bool Ipopt::GradientScaling::InitializeImpl (const OptionsList & options, const std::string & prefix)`
[protected],[virtual]

Initialize the object from the options.

Reimplemented from [Ipopt::StandardScalingBase](#).

6.50.3.3 `virtual void Ipopt::GradientScaling::DetermineScalingParametersImpl (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, const Matrix & Px_L, const Vector & x_L, const Matrix & Px_U, const Vector & x_U, Number & df, SmartPtr< Vector > & dx, SmartPtr< Vector > & dc, SmartPtr< Vector > & dd)` [protected],[virtual]

This is the method that has to be overloaded by a particular scaling method that somehow computes the scaling vectors dx, dc, and dd.

The pointers to those vectors can be NULL, in which case no scaling for that item will be done later.

Implements [Ipopt::StandardScalingBase](#).

6.50.3.4 `void Ipopt::GradientScaling::operator= (const GradientScaling &)` [private]

Overloaded Equals Operator.

6.50.4 Member Data Documentation

6.50.4.1 `SmartPtr<NLP> Ipopt::GradientScaling::nlp_` [private]

pointer to the [NLP](#) to get scaling parameters

Definition at line 81 of file IpGradientScaling.hpp.

6.50.4.2 `Number Ipopt::GradientScaling::scaling_max_gradient_` [private]

maximum allowed gradient before scaling is performed

Definition at line 84 of file IpGradientScaling.hpp.

6.50.4.3 `Number Ipopt::GradientScaling::scaling_obj_target_gradient_` [private]

target size of norm for objective gradient

Definition at line 87 of file IpGradientScaling.hpp.

6.50.4.4 `Number Ipopt::GradientScaling::scaling_constr_target_gradient_` [private]

target size of norm for constraint gradients

Definition at line 90 of file IpGradientScaling.hpp.

6.50.4.5 Number Ipopt::GradientScaling::scaling_min_value_ [private]

minimum value of a scaling parameter

Definition at line 93 of file IpGradientScaling.hpp.

The documentation for this class was generated from the following file:

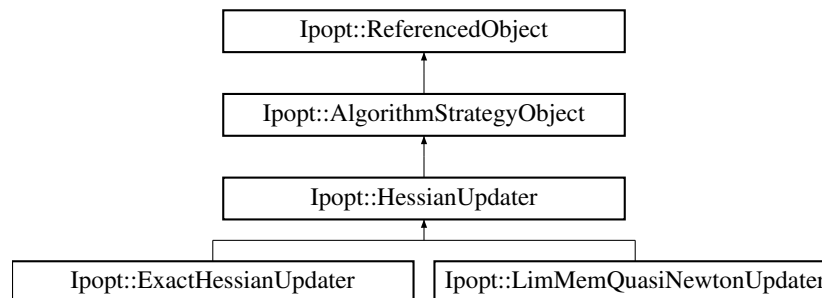
- Algorithm/IpGradientScaling.hpp

6.51 Ipopt::HessianUpdater Class Reference

Abstract base class for objects responsible for updating the Hessian information.

```
#include <IpHessianUpdater.hpp>
```

Inheritance diagram for Ipopt::HessianUpdater:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)
- virtual void [UpdateHessian](#) ()=0
Update the Hessian based on the current information in IpData, and possibly on information from previous calls.

Constructors/Destructors

- [HessianUpdater](#) ()
Default Constructor.
- virtual [~HessianUpdater](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [HessianUpdater](#) (const [HessianUpdater](#) &)
Copy Constructor.
- void [operator=](#) (const [HessianUpdater](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.51.1 Detailed Description

Abstract base class for objects responsible for updating the Hessian information.

This can be done using exact second derivatives from the [NLP](#), or by a quasi-Newton Option. The result is put into the W field in IpData.

Definition at line 22 of file IpHessianUpdater.hpp.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 Ipopt::HessianUpdater::HessianUpdater () [inline]

Default Constructor.

Definition at line 28 of file IpHessianUpdater.hpp.

6.51.2.2 virtual Ipopt::HessianUpdater::~~HessianUpdater () [inline],[virtual]

Default destructor.

Definition at line 32 of file IpHessianUpdater.hpp.

6.51.2.3 Ipopt::HessianUpdater::HessianUpdater (const HessianUpdater &) [private]

Copy Constructor.

6.51.3 Member Function Documentation

6.51.3.1 virtual bool Ipopt::HessianUpdater::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::LimMemQuasiNewtonUpdater](#), and [Ipopt::ExactHessianUpdater](#).

6.51.3.2 virtual void Ipopt::HessianUpdater::UpdateHessian () [pure virtual]

Update the Hessian based on the current information in IpData, and possibly on information from previous calls.

Implemented in [Ipopt::LimMemQuasiNewtonUpdater](#), and [Ipopt::ExactHessianUpdater](#).

6.51.3.3 void Ipopt::HessianUpdater::operator= (const HessianUpdater &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

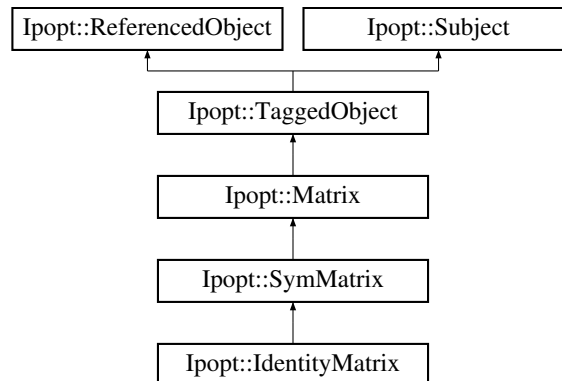
- [Algorithm/IpHessianUpdater.hpp](#)

6.52 Ipopt::IdentityMatrix Class Reference

Class for Matrices which are multiples of the identity matrix.

```
#include <IpIdentityMatrix.hpp>
```

Inheritance diagram for Ipopt::IdentityMatrix:



Public Member Functions

- void [SetFactor](#) ([Number](#) factor)
Method for setting the factor for the identity matrix.
- [Number](#) [GetFactor](#) () const
Method for getting the factor for the identity matrix.
- [Index](#) [Dim](#) () const
Method for obtaining the dimension of the matrix.

Constructors / Destructors

- [IdentityMatrix](#) (const [SymMatrixSpace](#) *owner_space)
Constructor, initializing with dimensions of the matrix (true identity matrix).
- [~IdentityMatrix](#) ()
Destructor.

Protected Member Functions

Methods overloaded from matrix

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual void [AddMSinvZImpl](#) ([Number](#) alpha, const [Vector](#) &S, const [Vector](#) &Z, [Vector](#) &X) const
 $X = X + \alpha * (\text{Matrix } S^{-1} \{ -1 \} Z).$
- virtual bool [HasValidNumbersImpl](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `IdentityMatrix ()`
Default Constructor.
- `IdentityMatrix (const IdentityMatrix &)`
Copy Constructor.
- `void operator= (const IdentityMatrix &)`
Overloaded Equals Operator.

Private Attributes

- `Number factor_`
Scaling factor for this identity matrix.

Additional Inherited Members

6.52.1 Detailed Description

Class for Matrices which are multiples of the identity matrix.

Definition at line 21 of file `IpIdentityMatrix.hpp`.

6.52.2 Constructor & Destructor Documentation

6.52.2.1 `Ipopt::IdentityMatrix::IdentityMatrix (const SymMatrixSpace * owner_space)`

Constructor, initializing with dimensions of the matrix (true identity matrix).

6.52.2.2 `Ipopt::IdentityMatrix::~~IdentityMatrix ()`

Destructor.

6.52.2.3 `Ipopt::IdentityMatrix::IdentityMatrix () [private]`

Default Constructor.

6.52.2.4 `Ipopt::IdentityMatrix::IdentityMatrix (const IdentityMatrix &) [private]`

Copy Constructor.

6.52.3 Member Function Documentation

6.52.3.1 `void Ipopt::IdentityMatrix::SetFactor (Number factor) [inline]`

Method for setting the factor for the identity matrix.

Definition at line 38 of file `IpIdentityMatrix.hpp`.

6.52.3.2 `Number Ipopt::IdentityMatrix::GetFactor () const [inline]`

Method for getting the factor for the identity matrix.

Definition at line 44 of file IpIdentityMatrix.hpp.

6.52.3.3 `Index Ipopt::IdentityMatrix::Dim () const`

Method for obtaining the dimension of the matrix.

6.52.3.4 `virtual void Ipopt::IdentityMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.52.3.5 `virtual void Ipopt::IdentityMatrix::AddMSinvZImpl (Number alpha, const Vector & S, const Vector & Z, Vector & X) const [protected], [virtual]`

$X = X + \alpha * (\text{Matrix } S^{-1} Z)$.

Prototype for this specialize method is provided, but for efficient implementation it should be overloaded for the expansion matrix.

Reimplemented from [Ipopt::Matrix](#).

6.52.3.6 `virtual bool Ipopt::IdentityMatrix::IsValidNumbersImpl () const [protected], [virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.52.3.7 `virtual void Ipopt::IdentityMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const [protected], [virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in *rows_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.52.3.8 `virtual void Ipopt::IdentityMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const [protected], [virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.52.3.9 `void Ipopt::IdentityMatrix::operator= (const IdentityMatrix &) [private]`

Overloaded Equals Operator.

6.52.4 Member Data Documentation**6.52.4.1** `Number Ipopt::IdentityMatrix::factor_ [private]`

Scaling factor for this identity matrix.

Definition at line 95 of file IpIdentityMatrix.hpp.

The documentation for this class was generated from the following file:

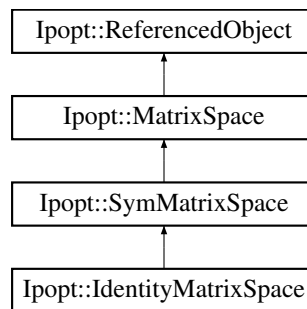
- [LinAlg/IpIdentityMatrix.hpp](#)

6.53 Ipopt::IdentityMatrixSpace Class Reference

This is the matrix space for [IdentityMatrix](#).

```
#include <IpIdentityMatrix.hpp>
```

Inheritance diagram for Ipopt::IdentityMatrixSpace:



Public Member Functions

- virtual [SymMatrix](#) * [MakeNewSymMatrix](#) () const
Overloaded MakeNew method for the [SymMatrixSpace](#) base class.
- [IdentityMatrix](#) * [MakeNewIdentityMatrix](#) () const
Method for creating a new matrix of this specific type.

Constructors / Destructors

- [IdentityMatrixSpace](#) ([Index](#) dim)
Constructor, given the dimension of the matrix.
- virtual [~IdentityMatrixSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IdentityMatrixSpace](#) ()
Default Constructor.
- [IdentityMatrixSpace](#) (const [IdentityMatrixSpace](#) &)
Copy Constructor.
- void [operator=](#) (const [IdentityMatrixSpace](#) &)
Overloaded Equals Operator.

6.53.1 Detailed Description

This is the matrix space for [IdentityMatrix](#).

Definition at line 99 of file IpIdentityMatrix.hpp.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 Ipopt::IdentityMatrixSpace::IdentityMatrixSpace (Index *dim*) [inline]

Constructor, given the dimension of the matrix.

Definition at line 105 of file IpIdentityMatrix.hpp.

6.53.2.2 virtual Ipopt::IdentityMatrixSpace::~~IdentityMatrixSpace () [inline],[virtual]

Destructor.

Definition at line 111 of file IpIdentityMatrix.hpp.

6.53.2.3 Ipopt::IdentityMatrixSpace::IdentityMatrixSpace () [private]

Default Constructor.

6.53.2.4 Ipopt::IdentityMatrixSpace::IdentityMatrixSpace (const IdentityMatrixSpace &) [private]

Copy Constructor.

6.53.3 Member Function Documentation

6.53.3.1 virtual SymMatrix* Ipopt::IdentityMatrixSpace::MakeNewSymMatrix () const [inline],[virtual]

Overloaded MakeNew method for the [SymMatrixSpace](#) base class.

Implements [Ipopt::SymMatrixSpace](#).

Definition at line 117 of file IpIdentityMatrix.hpp.

6.53.3.2 IdentityMatrix* Ipopt::IdentityMatrixSpace::MakeNewIdentityMatrix () const [inline]

Method for creating a new matrix of this specific type.

Definition at line 123 of file IpIdentityMatrix.hpp.

6.53.3.3 void Ipopt::IdentityMatrixSpace::operator= (const IdentityMatrixSpace &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

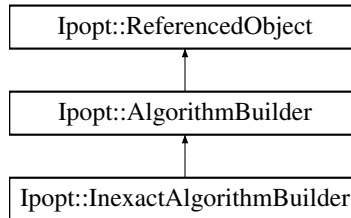
- [LinAlg/IpIdentityMatrix.hpp](#)

6.54 Ipopt::InexactAlgorithmBuilder Class Reference

Builder to create a complete IpoptAlg object for the inexact step computation version.

```
#include <IpInexactAlgBuilder.hpp>
```

Inheritance diagram for Ipopt::InexactAlgorithmBuilder:



Public Member Functions

Constructors/Destructors

- [InexactAlgorithmBuilder](#) ()
Constructor.
- virtual [~InexactAlgorithmBuilder](#) ()
Destructor.

Methods to build parts of the algorithm

- virtual void [BuildIpoptObjects](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix, const [SmartPtr](#)< [NLP](#) > &nlp, [SmartPtr](#)< [IpoptNLP](#) > &ip_nlp, [SmartPtr](#)< [IpoptData](#) > &ip_data, [SmartPtr](#)< [IpoptCalculatedQuantities](#) > &ip_cq)
- virtual [SmartPtr](#)< [IpoptAlgorithm](#) > [BuildBasicAlgorithm](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptTypeInfo.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [InexactAlgorithmBuilder](#) (const [InexactAlgorithmBuilder](#) &)
Default Constructor.
- void [operator=](#) (const [InexactAlgorithmBuilder](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [AugSystemSolver](#) > custom_solver_
Optional pointer to [AugSystemSolver](#).

6.54.1 Detailed Description

Builder to create a complete IpoptAlg object for the inexact step computation version.

Definition at line 21 of file IpInexactAlgBuilder.hpp.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 Ipopt::InexactAlgorithmBuilder::InexactAlgorithmBuilder ()

Constructor.

6.54.2.2 virtual Ipopt::InexactAlgorithmBuilder::~~InexactAlgorithmBuilder () [inline],[virtual]

Destructor.

Definition at line 30 of file IpInexactAlgBuilder.hpp.

6.54.2.3 Ipopt::InexactAlgorithmBuilder::InexactAlgorithmBuilder (const InexactAlgorithmBuilder &) [private]

Default Constructor.

Copy Constructor

6.54.3 Member Function Documentation

6.54.3.1 virtual void Ipopt::InexactAlgorithmBuilder::BuildIpoptObjects (const Journalist & jnlst, const OptionsList & options, const std::string & prefix, const SmartPtr< NLP > & nlp, SmartPtr< IpoptNLP > & ip_nlp, SmartPtr< IpoptData > & ip_data, SmartPtr< IpoptCalculatedQuantities > & ip_cq) [virtual]

Reimplemented from [Ipopt::AlgorithmBuilder](#).

6.54.3.2 virtual SmartPtr<IpoptAlgorithm> Ipopt::InexactAlgorithmBuilder::BuildBasicAlgorithm (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [virtual]

Reimplemented from [Ipopt::AlgorithmBuilder](#).

6.54.3.3 static void Ipopt::InexactAlgorithmBuilder::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]

Methods for IpoptTypeInfo.

register the options used by the algorithm builder

6.54.3.4 void Ipopt::InexactAlgorithmBuilder::operator= (const InexactAlgorithmBuilder &) [private]

Overloaded Equals Operator.

6.54.4 Member Data Documentation

6.54.4.1 SmartPtr<AugSystemSolver> Ipopt::InexactAlgorithmBuilder::custom_solver_ [private]

Optional pointer to [AugSystemSolver](#).

If this is set in the constructor, we will use this to solve the linear systems if the option linear_solver=custerm is chosen.

Definition at line 78 of file IpInexactAlgBuilder.hpp.

The documentation for this class was generated from the following file:

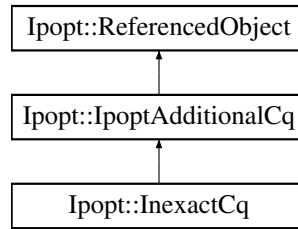
- Algorithm/Inexact/[IpInexactAlgBuilder.hpp](#)

6.55 Ipopt::InexactCq Class Reference

Class for all Chen-Goldfarb penalty method specific calculated quantities.

```
#include <IpInexactCq.hpp>
```

Inheritance diagram for Ipopt::InexactCq:



Public Member Functions

- bool [Initialize](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)
This method must be called to initialize the global algorithmic parameters.
- [SmartPtr](#)< const [Vector](#) > [curr_jac_cdT_times_curr_cdminuss](#) ()
Gradient of infeasibility w.r.t.
- [SmartPtr](#)< const [Vector](#) > [curr_scaling_slacks](#) ()
Vector of all inequality slacks for doing the slack-based scaling.
- [SmartPtr](#)< const [Vector](#) > [curr_slack_scaled_d_minus_s](#) ()
Vector with the slack-scaled d minus s inequalities.
- [Number](#) [curr_scaled_Ac_norm](#) ()
Scaled norm of Ac.
- [Number](#) [curr_scaled_A_norm2](#) ()
Scaled, squared norm of A.
- [Number](#) [slack_scaled_norm](#) (const [Vector](#) &x, const [Vector](#) &s)
Compute the 2-norm of a slack-scaled vector with x and s component.
- [SmartPtr](#)< const [Vector](#) > [curr_W_times_vec_x](#) (const [Vector](#) &vec_x)
*Compute x component of the W*vec product for the current Hessian and a vector.*
- [SmartPtr](#)< const [Vector](#) > [curr_W_times_vec_s](#) (const [Vector](#) &vec_s)
*Compute s component of the W*vec product for the current Hessian and a vector.*
- [SmartPtr](#)< const [Vector](#) > [curr_Wu_x](#) ()
*Compute x component of the W*u product for the current values.*
- [SmartPtr](#)< const [Vector](#) > [curr_Wu_s](#) ()
*Compute s component of the W*u product for the current values.*
- [Number](#) [curr_uWu](#) ()
Compute the $u^T W u$ product for the current values.
- [SmartPtr](#)< const [Vector](#) > [curr_jac_times_normal_c](#) ()
Compute the c-component of the product of the current constraint Jacobian with the current normal step.

- [SmartPtr](#)< const [Vector](#) > [curr_jac_times_normal_d](#) ()

Compute the d-component of the product of the current constraint Jacobian with the current normal step.

Constructors/Destructors

- [InexactCq](#) ([IpoptNLP](#) *ip_nlp, [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)

Constructor.

- virtual [~InexactCq](#) ()

Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)

Methods for IpoptType.

Private Member Functions

- [InexactData](#) & [InexData](#) ()

Method to easily access Inexact data.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [InexactCq](#) ()
Default Constructor.
- [InexactCq](#) (const [InexactCq](#) &)
Copy Constructor.
- void [operator=](#) (const [InexactCq](#) &)
Overloaded Equals Operator.

Private Attributes

- [Number](#) [slack_scale_max_](#)
Upper bound on slack-based scaling factors.

Pointers for easy access to data and NLP information. To

avoid circular references of Smart Pointers, we use a regular pointer here.

- [IpoptNLP](#) * [ip_nlp_](#)
- [IpoptData](#) * [ip_data_](#)
- [IpoptCalculatedQuantities](#) * [ip_cq_](#)

Caches

- [CachedResults](#)< [SmartPtr](#)< const [Vector](#) > > [curr_jac_cdT_times_curr_cdminuss_cache_](#)
- [CachedResults](#)< [SmartPtr](#)< const [Vector](#) > > [curr_scaling_slacks_cache_](#)

- [CachedResults< SmartPtr< const Vector > > curr_slack_scaled_d_minus_s_cache_](#)
- [CachedResults< Number > curr_scaled_Ac_norm_cache_](#)
- [CachedResults< Number > slack_scaled_norm_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_W_times_vec_x_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_W_times_vec_s_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_Wu_x_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_Wu_s_cache_](#)
- [CachedResults< Number > curr_uWu_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_jac_times_normal_c_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_jac_times_normal_d_cache_](#)

6.55.1 Detailed Description

Class for all Chen-Goldfarb penalty method specific calculated quantities.

Definition at line 22 of file `IpInexactCq.hpp`.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 `Ipopt::InexactCq::InexactCq (IpoptNLP * ip_nlp, IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq)`

Constructor.

6.55.2.2 `virtual Ipopt::InexactCq::~InexactCq () [virtual]`

Default destructor.

6.55.2.3 `Ipopt::InexactCq::InexactCq () [private]`

Default Constructor.

6.55.2.4 `Ipopt::InexactCq::InexactCq (const InexactCq &) [private]`

Copy Constructor.

6.55.3 Member Function Documentation

6.55.3.1 `bool Ipopt::InexactCq::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [virtual]`

This method must be called to initialize the global algorithmic parameters.

The parameters are taken from the [OptionsList](#) object.

Implements [Ipopt::IpoptAdditionalCq](#).

6.55.3.2 `static void Ipopt::InexactCq::RegisterOptions (const SmartPtr< RegisteredOptions > & roptions) [static]`

Methods for `IpoptType`.

6.55.3.3 SmartPtr<const Vector> Ipopt::InexactCq::curr_jac_cdT_times_curr_cdminuss ()

Gradient of infeasibility w.r.t.

x. Jacobian of equality constraints transpose times the equality constraints plus Jacobian of the inequality constraints transpose times the inequality constraints (including slacks).

6.55.3.4 SmartPtr<const Vector> Ipopt::InexactCq::curr_scaling_slacks ()

Vector of all inequality slacks for doing the slack-based scaling.

6.55.3.5 SmartPtr<const Vector> Ipopt::InexactCq::curr_slack_scaled_d_minus_s ()

Vector with the slack-scaled d minus s inequalities.

6.55.3.6 Number Ipopt::InexactCq::curr_scaled_Ac_norm ()

Scaled norm of Ac.

6.55.3.7 Number Ipopt::InexactCq::curr_scaled_A_norm2 ()

Scaled, squared norm of A.

6.55.3.8 Number Ipopt::InexactCq::slack_scaled_norm (const Vector & x, const Vector & s)

Compute the 2-norm of a slack-scaled vector with x and s component.

6.55.3.9 SmartPtr<const Vector> Ipopt::InexactCq::curr_W_times_vec_x (const Vector & vec_x)

Compute x component of the W*vec product for the current Hessian and a vector.

6.55.3.10 SmartPtr<const Vector> Ipopt::InexactCq::curr_W_times_vec_s (const Vector & vec_s)

Compute s component of the W*vec product for the current Hessian and a vector.

6.55.3.11 SmartPtr<const Vector> Ipopt::InexactCq::curr_Wu_x ()

Compute x component of the W*u product for the current values.

u here is the tangential step.

6.55.3.12 SmartPtr<const Vector> Ipopt::InexactCq::curr_Wu_s ()

Compute s component of the W*u product for the current values.

u here is the tangential step.

6.55.3.13 Number Ipopt::InexactCq::curr_uWu ()

Compute the $u^T W u$ product for the current values.

u here is the tangential step.

6.55.3.14 SmartPtr<const Vector> Ipopt::InexactCq::curr_jac_times_normal_c ()

Compute the c-component of the product of the current constraint Jacobian with the current normal step.

6.55.3.15 SmartPtr<const Vector> Ipopt::InexactCq::curr_jac_times_normal_d ()

Compute the d-component of the product of the current constraint Jacobian with the current normal step.

6.55.3.16 `void Ipopt::InexactCq::operator= (const InexactCq &) [private]`

Overloaded Equals Operator.

6.55.3.17 `InexactData& Ipopt::InexactCq::InexData () [inline],[private]`

Method to easily access Inexact data.

Definition at line 128 of file `IpInexactCq.hpp`.

6.55.4 Member Data Documentation

6.55.4.1 `IpoptNLP* Ipopt::InexactCq::ip_nlp_ [private]`

Definition at line 122 of file `IpInexactCq.hpp`.

6.55.4.2 `IpoptData* Ipopt::InexactCq::ip_data_ [private]`

Definition at line 123 of file `IpInexactCq.hpp`.

6.55.4.3 `IpoptCalculatedQuantities* Ipopt::InexactCq::ip_cq_ [private]`

Definition at line 124 of file `IpInexactCq.hpp`.

6.55.4.4 `CachedResults<SmartPtr<const Vector> > Ipopt::InexactCq::curr_jac_cdT_times_curr_cdminuss_cache_ [private]`

Definition at line 138 of file `IpInexactCq.hpp`.

6.55.4.5 `CachedResults<SmartPtr<const Vector> > Ipopt::InexactCq::curr_scaling_slacks_cache_ [private]`

Definition at line 139 of file `IpInexactCq.hpp`.

6.55.4.6 `CachedResults<SmartPtr<const Vector> > Ipopt::InexactCq::curr_slack_scaled_d_minus_s_cache_ [private]`

Definition at line 140 of file `IpInexactCq.hpp`.

6.55.4.7 `CachedResults<Number> Ipopt::InexactCq::curr_scaled_Ac_norm_cache_ [private]`

Definition at line 141 of file `IpInexactCq.hpp`.

6.55.4.8 `CachedResults<Number> Ipopt::InexactCq::slack_scaled_norm_cache_ [private]`

Definition at line 142 of file `IpInexactCq.hpp`.

6.55.4.9 `CachedResults<SmartPtr<const Vector> > Ipopt::InexactCq::curr_W_times_vec_x_cache_ [private]`

Definition at line 143 of file `IpInexactCq.hpp`.

6.55.4.10 `CachedResults<SmartPtr<const Vector> > Ipopt::InexactCq::curr_W_times_vec_s_cache_ [private]`

Definition at line 144 of file `IpInexactCq.hpp`.

6.55.4.11 `CachedResults<SmartPtr<const Vector> > Ipopt::InexactCq::curr_Wu_x_cache_ [private]`

Definition at line 145 of file `IpInexactCq.hpp`.

6.55.4.12 **CachedResults**<**SmartPtr**<const **Vector**> > **Ipopt::InexactCq::curr_Wu_s_cache_** [private]

Definition at line 146 of file IpInexactCq.hpp.

6.55.4.13 **CachedResults**<**Number**> **Ipopt::InexactCq::curr_uWu_cache_** [private]

Definition at line 147 of file IpInexactCq.hpp.

6.55.4.14 **CachedResults**<**SmartPtr**<const **Vector**> > **Ipopt::InexactCq::curr_jac_times_normal_c_cache_** [private]

Definition at line 148 of file IpInexactCq.hpp.

6.55.4.15 **CachedResults**<**SmartPtr**<const **Vector**> > **Ipopt::InexactCq::curr_jac_times_normal_d_cache_** [private]

Definition at line 149 of file IpInexactCq.hpp.

6.55.4.16 **Number** **Ipopt::InexactCq::slack_scale_max_** [private]

Upper bound on slack-based scaling factors.

Definition at line 153 of file IpInexactCq.hpp.

The documentation for this class was generated from the following file:

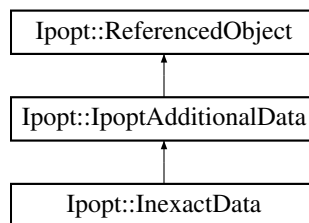
- Algorithm/Inexact/[IpInexactCq.hpp](#)

6.56 Ipopt::InexactData Class Reference

Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm.

```
#include <IpInexactData.hpp>
```

Inheritance diagram for Ipopt::InexactData:



Public Member Functions

Constructors/Destructors

- [InexactData](#) ()
Constructor.
- [~InexactData](#) ()
Default destructor.

Methods overloaded from IpoptAdditionalData

- bool [Initialize](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)
This method must be called to initialize the global algorithmic parameters.

- bool `InitializeDataStructures` ()
Initialize Data Structures at the beginning.
- void `AcceptTrialPoint` ()
Do whatever is necessary to accept a trial point as current iterate.

Normal step set and accessor methods

- void `set_normal_x` (SmartPtr< Vector > &normal_x)
- void `set_normal_s` (SmartPtr< Vector > &normal_s)
- SmartPtr< const Vector > `normal_x` ()
- SmartPtr< const Vector > `normal_s` ()

Tangential step set and accessor methods

- void `set_tangential_x` (SmartPtr< const Vector > &tangential_x)
- void `set_tangential_s` (SmartPtr< const Vector > &tangential_s)
- SmartPtr< const Vector > `tangential_x` ()
- SmartPtr< const Vector > `tangential_s` ()

Flag indicating if most recent step has been fully

accepted.

This is used to determine if the trust region radius should be increased.

- void `set_full_step_accepted` (bool full_step_accepted)
- bool `full_step_accepted` ()

Current value of penalty parameter

- void `set_curr_nu` (Number nu)
- Number `curr_nu` ()

Current normal step computation flag

- void `set_compute_normal` (bool compute_normal)
- bool `compute_normal` ()

Next iteration normal step computation flag

- void `set_next_compute_normal` (bool next_compute_normal)
- bool `next_compute_normal` ()

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `InexactData` (const `InexactData` &)
Copy Constructor.
- void `operator=` (const `InexactData` &)
Overloaded Equals Operator.

Private Attributes

- bool [full_step_accepted_](#)
Flag indicating if most recent step has been fully accepted.
- Number [curr_nu_](#)
current value of penalty parameter
- bool [compute_normal_](#)
current normal step computation flag
- bool [next_compute_normal_](#)
next iteration normal step computation flag

Normal step

- [SmartPtr](#)< const [Vector](#) > [normal_x_](#)
- [SmartPtr](#)< const [Vector](#) > [normal_s_](#)

Tangential step

- [SmartPtr](#)< const [Vector](#) > [tangential_x_](#)
- [SmartPtr](#)< const [Vector](#) > [tangential_s_](#)

6.56.1 Detailed Description

Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm.
Definition at line 19 of file `IpInexactData.hpp`.

6.56.2 Constructor & Destructor Documentation

6.56.2.1 Ipopt::InexactData::InexactData ()

Constructor.

6.56.2.2 Ipopt::InexactData::~~InexactData ()

Default destructor.

6.56.2.3 Ipopt::InexactData::InexactData (const InexactData &) [private]

Copy Constructor.

6.56.3 Member Function Documentation

6.56.3.1 bool Ipopt::InexactData::Initialize (const Journalist & *jnlst*, const OptionsList & *options*, const std::string & *prefix*) [virtual]

This method must be called to initialize the global algorithmic parameters.

The parameters are taken from the [OptionsList](#) object.

Implements [Ipopt::IpoptAdditionalData](#).

6.56.3.2 `bool Ipopt::InexactData::InitializeDataStructures () [virtual]`

Initialize Data Structures at the beginning.

Implements [Ipopt::IpoptAdditionalData](#).

6.56.3.3 `void Ipopt::InexactData::AcceptTrialPoint () [virtual]`

Do whatever is necessary to accept a trial point as current iterate.

This is also used to finish an iteration, i.e., to release memory, and to reset any flags for a new iteration.

Implements [Ipopt::IpoptAdditionalData](#).

6.56.3.4 `void Ipopt::InexactData::set_normal_x (SmartPtr< Vector > & normal_x) [inline]`

Definition at line 51 of file `IpInexactData.hpp`.

6.56.3.5 `void Ipopt::InexactData::set_normal_s (SmartPtr< Vector > & normal_s) [inline]`

Definition at line 56 of file `IpInexactData.hpp`.

6.56.3.6 `SmartPtr<const Vector> Ipopt::InexactData::normal_x () [inline]`

Definition at line 61 of file `IpInexactData.hpp`.

6.56.3.7 `SmartPtr<const Vector> Ipopt::InexactData::normal_s () [inline]`

Definition at line 65 of file `IpInexactData.hpp`.

6.56.3.8 `void Ipopt::InexactData::set_tangential_x (SmartPtr< const Vector > & tangential_x) [inline]`

Definition at line 73 of file `IpInexactData.hpp`.

6.56.3.9 `void Ipopt::InexactData::set_tangential_s (SmartPtr< const Vector > & tangential_s) [inline]`

Definition at line 78 of file `IpInexactData.hpp`.

6.56.3.10 `SmartPtr<const Vector> Ipopt::InexactData::tangential_x () [inline]`

Definition at line 83 of file `IpInexactData.hpp`.

6.56.3.11 `SmartPtr<const Vector> Ipopt::InexactData::tangential_s () [inline]`

Definition at line 87 of file `IpInexactData.hpp`.

6.56.3.12 `void Ipopt::InexactData::set_full_step_accepted (bool full_step_accepted) [inline]`

Definition at line 97 of file `IpInexactData.hpp`.

6.56.3.13 `bool Ipopt::InexactData::full_step_accepted () [inline]`

Definition at line 101 of file `IpInexactData.hpp`.

6.56.3.14 `void Ipopt::InexactData::set_curr_nu (Number nu) [inline]`

Definition at line 109 of file `IpInexactData.hpp`.

6.56.3.15 **Number** Ipopt::InexactData::curr_nu () [inline]

Definition at line 113 of file IpInexactData.hpp.

6.56.3.16 **void** Ipopt::InexactData::set_compute_normal (*bool compute_normal*) [inline]

Definition at line 121 of file IpInexactData.hpp.

6.56.3.17 **bool** Ipopt::InexactData::compute_normal () [inline]

Definition at line 125 of file IpInexactData.hpp.

6.56.3.18 **void** Ipopt::InexactData::set_next_compute_normal (*bool next_compute_normal*) [inline]

Definition at line 133 of file IpInexactData.hpp.

6.56.3.19 **bool** Ipopt::InexactData::next_compute_normal () [inline]

Definition at line 137 of file IpInexactData.hpp.

6.56.3.20 **void** Ipopt::InexactData::operator= (*const InexactData &*) [private]

Overloaded Equals Operator.

6.56.4 Member Data Documentation

6.56.4.1 **SmartPtr<const Vector>** Ipopt::InexactData::normal_x_ [private]

Definition at line 161 of file IpInexactData.hpp.

6.56.4.2 **SmartPtr<const Vector>** Ipopt::InexactData::normal_s_ [private]

Definition at line 162 of file IpInexactData.hpp.

6.56.4.3 **SmartPtr<const Vector>** Ipopt::InexactData::tangential_x_ [private]

Definition at line 167 of file IpInexactData.hpp.

6.56.4.4 **SmartPtr<const Vector>** Ipopt::InexactData::tangential_s_ [private]

Definition at line 168 of file IpInexactData.hpp.

6.56.4.5 **bool** Ipopt::InexactData::full_step_accepted_ [private]

Flag indicating if most recent step has been fully accepted.

Definition at line 172 of file IpInexactData.hpp.

6.56.4.6 **Number** Ipopt::InexactData::curr_nu_ [private]

current value of penalty parameter

Definition at line 175 of file IpInexactData.hpp.

6.56.4.7 **bool** Ipopt::InexactData::compute_normal_ [private]

current normal step computation flag

Definition at line 178 of file IpInexactData.hpp.

6.56.4.8 `bool Ipopt::InexactData::next_compute_normal_ [private]`

next iteration normal step computation flag

Definition at line 181 of file IpInexactData.hpp.

The documentation for this class was generated from the following file:

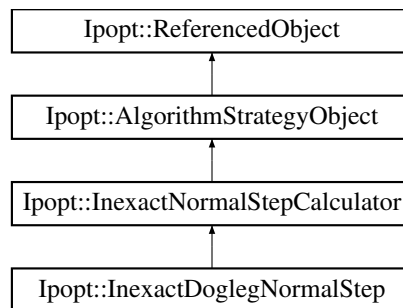
- Algorithm/Inexact/IpInexactData.hpp

6.57 Ipopt::InexactDoglegNormalStep Class Reference

Compute the normal step using a dogleg approach.

```
#include <IpInexactDoglegNormal.hpp>
```

Inheritance diagram for Ipopt::InexactDoglegNormalStep:



Public Member Functions

- virtual bool `InitializeImpl` (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool `ComputeNormalStep` ([SmartPtr](#)< [Vector](#) > &normal_x, [SmartPtr](#)< [Vector](#) > &normal_s)
Method for computing the normal step.

Constructors/Destructors

- `InexactDoglegNormalStep` ([SmartPtr](#)< [InexactNewtonNormalStep](#) > newton_step, [SmartPtr](#)< [InexactNormalTerminationTester](#) > normal_tester=NULL)
Default onstructor.
- virtual `~InexactDoglegNormalStep` ()
Default destructor.

Static Public Member Functions

- static void `RegisterOptions` ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [InexactDoglegNormalStep](#) ()
Default onstructor.
- [InexactDoglegNormalStep](#) (const [InexactDoglegNormalStep](#) &)
Copy Constructor.
- void [operator=](#) (const [InexactDoglegNormalStep](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [InexactNewtonNormalStep](#) > [newton_step_](#)
Pointer to object for computing the "Newton" step in the dogleg method.
- [SmartPtr](#)
< [InexactNormalTerminationTester](#) > [normal_tester_](#)
Pointer to object that is used by the newton_step computation object to determine if iterative solver is done.
- [Number](#) [curr_omega_](#)
Current value of the trust region factor.
- bool [last_tr_inactive_](#)
Flag indicating if trust region was active in last iteration.

Algorithmic options

- [Number](#) [omega_max_](#)

Additional Inherited Members

6.57.1 Detailed Description

Compute the normal step using a dogleg approach.

Definition at line 20 of file `IpInexactDoglegNormal.hpp`.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 `Ipopt::InexactDoglegNormalStep::InexactDoglegNormalStep (SmartPtr< InexactNewtonNormalStep > newton_step, SmartPtr< InexactNormalTerminationTester > normal_tester = NULL)`

Default onstructor.

6.57.2.2 `virtual Ipopt::InexactDoglegNormalStep::~~InexactDoglegNormalStep () [virtual]`

Default destructor.

6.57.2.3 `Ipopt::InexactDoglegNormalStep::InexactDoglegNormalStep () [private]`

Default onstructor.

6.57.2.4 `Ipopt::InexactDoglegNormalStep::InexactDoglegNormalStep (const InexactDoglegNormalStep &) [private]`

Copy Constructor.

6.57.3 Member Function Documentation

6.57.3.1 `virtual bool Ipopt::InexactDoglegNormalStep::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::InexactNormalStepCalculator](#).

6.57.3.2 `virtual bool Ipopt::InexactDoglegNormalStep::ComputeNormalStep (SmartPtr< Vector > & normal_x, SmartPtr< Vector > & normal_s) [virtual]`

Method for computing the normal step.

The computed step is returned as `normal_x` and `normal_s`, for the `x` and `s` variables, respectively. These quantities are not slack-scaled. If the step cannot be computed, this method returns false.

Implements [Ipopt::InexactNormalStepCalculator](#).

6.57.3.3 `static void Ipopt::InexactDoglegNormalStep::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for `IpoptType`.

6.57.3.4 `void Ipopt::InexactDoglegNormalStep::operator= (const InexactDoglegNormalStep &) [private]`

Overloaded Equals Operator.

6.57.4 Member Data Documentation

6.57.4.1 `SmartPtr<InexactNewtonNormalStep> Ipopt::InexactDoglegNormalStep::newton_step_ [private]`

Pointer to object for computing the "Newton" step in the dogleg method.

Definition at line 70 of file `IpInexactDoglegNormal.hpp`.

6.57.4.2 `SmartPtr<InexactNormalTerminationTester> Ipopt::InexactDoglegNormalStep::normal_tester_ [private]`

Pointer to object that is used by the `newton_step` computation object to determine if iterative solver is done.

This is needed here because this dogleg object is setting the value of the linearized constraint violation at the cauchy point if `normal_tester` is not NULL.

Definition at line 77 of file `IpInexactDoglegNormal.hpp`.

6.57.4.3 `Number Ipopt::InexactDoglegNormalStep::omega_max_ [private]`

Definition at line 81 of file `IpInexactDoglegNormal.hpp`.

6.57.4.4 `Number Ipopt::InexactDoglegNormalStep::curr_omega_ [private]`

Current value of the trust region factor.

Definition at line 85 of file `IpInexactDoglegNormal.hpp`.

6.57.4.5 bool Ipopt::InexactDoglegNormalStep::last_tr_inactive_ [private]

Flag indicating if trust region was active in last iteration.

Definition at line 88 of file IpInexactDoglegNormal.hpp.

The documentation for this class was generated from the following file:

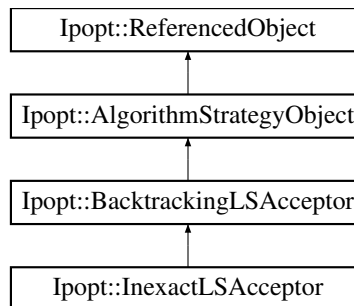
- Algorithm/Inexact/IpInexactDoglegNormal.hpp

6.58 Ipopt::InexactLSAcceptor Class Reference

Penalty function line search for the inexact step algorithm version.

```
#include <IpInexactLSAcceptor.hpp>
```

Inheritance diagram for Ipopt::InexactLSAcceptor:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
InitializeImpl - overloaded from [AlgorithmStrategyObject](#).
- virtual void [Reset](#) ()
Reset the acceptor.
- virtual void [InitThisLineSearch](#) (bool in_watchdog)
Initialization for the next line search.
- virtual void [PrepareRestoPhaseStart](#) ()
Method that is called before the restoration phase is called.
- virtual [Number](#) [CalculateAlphaMin](#) ()
Method returning the lower bound on the trial step sizes.
- virtual bool [CheckAcceptabilityOfTrialPoint](#) ([Number](#) alpha_primal)
Method for checking if current trial point is acceptable.
- virtual bool [TrySecondOrderCorrection](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try a second order correction for the constraints.
- virtual bool [TryCorrector](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try higher order corrector (for fast local convergence).
- virtual char [UpdateForNextIteration](#) ([Number](#) alpha_primal_test)
Method for ending the current line search.

- virtual void [StartWatchDog](#) ()
Method for setting internal data if the watchdog procedure is started.
- virtual void [StopWatchDog](#) ()
Method for setting internal data if the watchdog procedure is stopped.
- virtual [Number](#) [ComputeAlphaForY](#) ([Number](#) alpha_primal, [Number](#) alpha_dual, [SmartPtr](#)< [IteratesVector](#) > &delta)
Method for updating the equality constraint multipliers.
- virtual bool [HasComputeAlphaForY](#) () const
Method returning true of ComputeAlphaForY is implemented for this acceptor.

Constructors/Destructors

- [InexactLSAcceptor](#) ()
Constructor.
- virtual [~InexactLSAcceptor](#) ()
Default destructor.

Trial Point Accepting Methods. Used internally to check certain

acceptability criteria and used externally (by the restoration phase convergence check object, for instance)

- bool [IsAcceptableToCurrentIterate](#) ([Number](#) trial_barr, [Number](#) trial_theta, bool called_from_restoration=false) const
Checks if a trial point is acceptable to the current iterate.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for [OptionsList](#).

Protected Member Functions

- [InexactData](#) & [InexData](#) ()
Method to easily access Inexact data.
- [InexactCq](#) & [InexCq](#) ()
Method to easily access Inexact calculated quantities.

Private Member Functions

- [Number](#) [CalcPred](#) ([Number](#) alpha)
Compute predicted reduction for given step size.
- void [ResetSlacks](#) ()
Method for resetting the slacks to be satisfying the slack equality constraints without increasing the barrier function

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [InexactLSAcceptor](#) (const [InexactLSAcceptor](#) &)
Copy Constructor.
- void [operator=](#) (const [InexactLSAcceptor](#) &)
Overloaded Equals Operator.

Private Attributes

- bool [in_tt2_](#)
Flag indicating if this is a termination test 2 iteration in which we just update the multipliers and skip the line search.
- [Number resto_pred_](#)
When called from the restoration phase, this is the required predicted reduction.
- bool [accepted_by_low_only_](#)
Flag indicating if the step was accepted only because of the lower penalty parameter.

Parameters for the penalty function algorithm.

- [Number nu_init_](#)
Initial value of penalty parameter.
- [Number nu_low_init_](#)
Initial value of lower penalty parameter.
- [Number nu_low_fact_](#)
Factor in update rule for lower penalty parameter.
- [Number nu_inc_](#)
Increment for penalty parameter.
- [Number eta_](#)
 η_ϕ
- [Number rho_](#)
 ρ
- [Number tcc_theta_](#)
theta factor in Tangential Component Condition
- [Number nu_update_inf_skip_tol_](#)
Lower feasibility bound to skip penalty parameter update.
- bool [flexible_penalty_function_](#)
Flag indicating whether the Curtis/Nocedal flexible penalty function should be used.

Information related to watchdog procedure

- [Number reference_theta_](#)
Constraint violation at the point with respect to which progress is to be made.
- [Number reference_barr_](#)
Barrier objective function at the point with respect to which progress is to be made.
- [Number reference_pred_](#)
Reference predicted reduction.
- [Number watchdog_theta_](#)
Constraint violation at reference point.
- [Number watchdog_barr_](#)
Barrier objective function at reference point.
- [Number watchdog_pred_](#)
Predicted reduction to be compared with in watch dog.

Penalty parameter

- [Number nu_](#)

- Current value of the penalty parameter.*
- [Number last_nu_](#)
Value of penalty parameter at beginning of the iteration.
- [Number nu_low_](#)
Current lower value of the penalty parameter.
- [Number last_nu_low_](#)
Value of lower penalty parameter at beginning of the iteration.
- [Number inexact_decomposition_activate_tol_](#)
Step size threshold for activating step decomposition.
- [Number inexact_decomposition_inactivate_tol_](#)
Step size threshold for inactivating step decomposition.

6.58.1 Detailed Description

Penalty function line search for the inexact step algorithm version.

Definition at line 22 of file `IpInexactLSAcceptor.hpp`.

6.58.2 Constructor & Destructor Documentation

6.58.2.1 `Ipopt::InexactLSAcceptor::InexactLSAcceptor ()`

Constructor.

The [PDSysolver](#) object only needs to be provided (i.e. not NULL) if second order correction or corrector steps are to be used.

6.58.2.2 `virtual Ipopt::InexactLSAcceptor::~~InexactLSAcceptor () [virtual]`

Default destructor.

6.58.2.3 `Ipopt::InexactLSAcceptor::InexactLSAcceptor (const InexactLSAcceptor &) [private]`

Copy Constructor.

6.58.3 Member Function Documentation

6.58.3.1 `virtual bool Ipopt::InexactLSAcceptor::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

InitializeImpl - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.2 `virtual void Ipopt::InexactLSAcceptor::Reset () [virtual]`

Reset the acceptor.

This function should be called if all previous information should be discarded when the line search is performed the next time. For example, this method should be called if the barrier parameter is changed.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.3 `virtual void Ipopt::InexactLSAcceptor::InitThisLineSearch (bool in_watchdog) [virtual]`

Initialization for the next line search.

The flag `in_watchdog` indicates if we are currently in an active watchdog procedure. Here is where the penalty parameter is updated.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.4 `virtual void Ipopt::InexactLSAcceptor::PrepareRestoPhaseStart () [virtual]`

Method that is called before the restoration phase is called.

For now, we just terminate if this is called.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.5 `virtual Number Ipopt::InexactLSAcceptor::CalculateAlphaMin () [virtual]`

Method returning the lower bound on the trial step sizes.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.6 `virtual bool Ipopt::InexactLSAcceptor::CheckAcceptabilityOfTrialPoint (Number alpha_primal) [virtual]`

Method for checking if current trial point is acceptable.

It is assumed that the delta information in `ip_data` is the search direction used in criteria. The primal trial point has to be set before the call.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.7 `virtual bool Ipopt::InexactLSAcceptor::TrySecondOrderCorrection (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta) [virtual]`

Try a second order correction for the constraints.

For the inexact version, this always returns false because a second order step is too expensive.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.8 `virtual bool Ipopt::InexactLSAcceptor::TryCorrector (Number alpha_primal_test, Number & alpha_primal, SmartPtr< IteratesVector > & actual_delta) [virtual]`

Try higher order corrector (for fast local convergence).

In contrast to a second order correction step, which tries to make an unacceptable point acceptable by improving constraint violation, this corrector step is tried even if the regular primal-dual step is acceptable.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.9 `virtual char Ipopt::InexactLSAcceptor::UpdateForNextIteration (Number alpha_primal_test) [virtual]`

Method for ending the current line search.

When it is called, the internal data should be updates. `alpha_primal_test` is the value of alpha that has been used for in the acceptance test ealier.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.10 `virtual void Ipopt::InexactLSAcceptor::StartWatchDog () [virtual]`

Method for setting internal data if the watchdog procedure is started.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.11 `virtual void Ipopt::InexactLSAcceptor::StopWatchDog () [virtual]`

Method for setting internal data if the watchdog procedure is stopped.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.12 `bool Ipopt::InexactLSAcceptor::IsAcceptableToCurrentIterate (Number trial_barr, Number trial_theta, bool called_from_restoration = false) const`

Checks if a trial point is acceptable to the current iterate.

6.58.3.13 `virtual Number Ipopt::InexactLSAcceptor::ComputeAlphaForY (Number alpha_primal, Number alpha_dual, SmartPtr< IteratesVector > & delta) [virtual]`

Method for updating the equality constraint multipliers.

Reimplemented from [Ipopt::BacktrackingLSAcceptor](#).

6.58.3.14 `virtual bool Ipopt::InexactLSAcceptor::HasComputeAlphaForY () const [inline],[virtual]`

Method returning true if ComputeAlphaForY is implemented for this acceptor.

Reimplemented from [Ipopt::BacktrackingLSAcceptor](#).

Definition at line 116 of file `IpInexactLSAcceptor.hpp`.

6.58.3.15 `static void Ipopt::InexactLSAcceptor::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for [OptionsList](#).

6.58.3.16 `InexactData& Ipopt::InexactLSAcceptor::InexData () [inline],[protected]`

Method to easily access Inexact data.

Definition at line 128 of file `IpInexactLSAcceptor.hpp`.

6.58.3.17 `InexactCq& Ipopt::InexactLSAcceptor::InexCq () [inline],[protected]`

Method to easily access Inexact calculated quantities.

Definition at line 137 of file `IpInexactLSAcceptor.hpp`.

6.58.3.18 `void Ipopt::InexactLSAcceptor::operator= (const InexactLSAcceptor &) [private]`

Overloaded Equals Operator.

6.58.3.19 `Number Ipopt::InexactLSAcceptor::CalcPred (Number alpha) [private]`

Compute predicted reduction for given step size.

6.58.3.20 `void Ipopt::InexactLSAcceptor::ResetSlacks () [private]`

Method for resetting the slacks to be satisfying the slack

equality constraints without increasing the barrier function

6.58.4 Member Data Documentation

6.58.4.1 Number Ipopt::InexactLSAcceptor::nu_init_ [private]

Initial value of penalty parameter.

Definition at line 172 of file IpInexactLSAcceptor.hpp.

6.58.4.2 Number Ipopt::InexactLSAcceptor::nu_low_init_ [private]

Initial value of lower penalty parameter.

Definition at line 174 of file IpInexactLSAcceptor.hpp.

6.58.4.3 Number Ipopt::InexactLSAcceptor::nu_low_fact_ [private]

Factor in update rule for lower penalty parameter.

Definition at line 176 of file IpInexactLSAcceptor.hpp.

6.58.4.4 Number Ipopt::InexactLSAcceptor::nu_inc_ [private]

Increment for penalty parameter.

Definition at line 178 of file IpInexactLSAcceptor.hpp.

6.58.4.5 Number Ipopt::InexactLSAcceptor::eta_ [private]

η_ϕ

Definition at line 180 of file IpInexactLSAcceptor.hpp.

6.58.4.6 Number Ipopt::InexactLSAcceptor::rho_ [private]

ρ

Definition at line 182 of file IpInexactLSAcceptor.hpp.

6.58.4.7 Number Ipopt::InexactLSAcceptor::tcc_theta_ [private]

theta factor in Tangential Component Condition

Definition at line 184 of file IpInexactLSAcceptor.hpp.

6.58.4.8 Number Ipopt::InexactLSAcceptor::nu_update_inf_skip_tol_ [private]

Lower feasibility bound to skip penalty parameter update.

Definition at line 186 of file IpInexactLSAcceptor.hpp.

6.58.4.9 bool Ipopt::InexactLSAcceptor::flexible_penalty_function_ [private]

Flag indicating whether the Curtis/Nocedal flexible penalty function should be used.

Definition at line 189 of file IpInexactLSAcceptor.hpp.

6.58.4.10 Number Ipopt::InexactLSAcceptor::reference_theta_ [private]

Constraint violation at the point with respect to which progress is to be made.

Definition at line 196 of file IpInexactLSAcceptor.hpp.

6.58.4.11 `Number Ipopt::InexactLSAcceptor::reference_barr_ [private]`

Barrier objective function at the point with respect to which progress is to be made.

Definition at line 199 of file IpInexactLSAcceptor.hpp.

6.58.4.12 `Number Ipopt::InexactLSAcceptor::reference_pred_ [private]`

Reference predicted reduction.

If positive, then it is used in watch dog.

Definition at line 202 of file IpInexactLSAcceptor.hpp.

6.58.4.13 `Number Ipopt::InexactLSAcceptor::watchdog_theta_ [private]`

Constraint violation at reference point.

Definition at line 204 of file IpInexactLSAcceptor.hpp.

6.58.4.14 `Number Ipopt::InexactLSAcceptor::watchdog_barr_ [private]`

Barrier objective function at reference point.

Definition at line 206 of file IpInexactLSAcceptor.hpp.

6.58.4.15 `Number Ipopt::InexactLSAcceptor::watchdog_pred_ [private]`

Predicted reduction to be compared with in watch dog.

Definition at line 208 of file IpInexactLSAcceptor.hpp.

6.58.4.16 `Number Ipopt::InexactLSAcceptor::nu_ [private]`

Current value of the penalty parameter.

Definition at line 214 of file IpInexactLSAcceptor.hpp.

6.58.4.17 `Number Ipopt::InexactLSAcceptor::last_nu_ [private]`

Value of penalty parameter at beginning of the iteration.

Definition at line 216 of file IpInexactLSAcceptor.hpp.

6.58.4.18 `Number Ipopt::InexactLSAcceptor::nu_low_ [private]`

Current lower value of the penalty parameter.

Definition at line 218 of file IpInexactLSAcceptor.hpp.

6.58.4.19 `Number Ipopt::InexactLSAcceptor::last_nu_low_ [private]`

Value of lower penalty parameter at beginning of the iteration.

Definition at line 220 of file IpInexactLSAcceptor.hpp.

6.58.4.20 `Number Ipopt::InexactLSAcceptor::inexact_decomposition_activate_tol_ [private]`

Step size threshold for activating step decomposition.

Definition at line 222 of file IpInexactLSAcceptor.hpp.

6.58.4.21 **Number** Ipopt::InexactLSAcceptor::inexact_decomposition_inactivate_tol_ [private]

Step size threshold for inactivating step decomposition.

Definition at line 224 of file IpInexactLSAcceptor.hpp.

6.58.4.22 **bool** Ipopt::InexactLSAcceptor::in_tt2_ [private]

Flag indicating if this is a termination test 2 iteration in which we just update the multipliers and skip the line search.

Definition at line 230 of file IpInexactLSAcceptor.hpp.

6.58.4.23 **Number** Ipopt::InexactLSAcceptor::resto_pred_ [private]

When called from the restoration phase, this is the required predicted reduction.

Definition at line 234 of file IpInexactLSAcceptor.hpp.

6.58.4.24 **bool** Ipopt::InexactLSAcceptor::accepted_by_low_only_ [private]

Flag indicating if the step was accepted only because of the lower penalty parameter.

This is for output only.

Definition at line 238 of file IpInexactLSAcceptor.hpp.

The documentation for this class was generated from the following file:

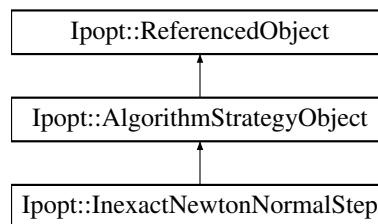
- Algorithm/Inexact/IpInexactLSAcceptor.hpp

6.59 Ipopt::InexactNewtonNormalStep Class Reference

Compute the "Newton" normal step from the (slack-scaled) augmented system.

```
#include <IpInexactNewtonNormal.hpp>
```

Inheritance diagram for Ipopt::InexactNewtonNormalStep:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [ComputeNewtonNormalStep](#) ([Vector](#) &newton_x, [Vector](#) &newton_s)
Method for computing the normal step.

Constructors/Destructors

- [InexactNewtonNormalStep](#) (SmartPtr< [AugSystemSolver](#) > aug_solver)

- *Default onstructor.*
- virtual `~InexactNewtonNormalStep ()`
- *Default destructor.*

Static Public Member Functions

- static void `RegisterOptions (SmartPtr< RegisteredOptions > roptions)`
- *Methods for IpoptType.*

Protected Member Functions

- `InexactData & InexData ()`
- *Method to easily access Inexact data.*
- `InexactCq & InexCq ()`
- *Method to easily access Inexact calculated quantities.*

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `InexactNewtonNormalStep ()`
- *Default onstructor.*
- `InexactNewtonNormalStep (const InexactNewtonNormalStep &)`
- *Copy Constructor.*
- void `operator= (const InexactNewtonNormalStep &)`
- *Overloaded Equals Operator.*

Private Attributes

- `SmartPtr< AugSystemSolver > aug_solver_`
- *Object to be used to solve the augmented system.*

6.59.1 Detailed Description

Compute the "Newton" normal step from the (slack-scaled) augmented system.

Definition at line 21 of file `IpInexactNewtonNormal.hpp`.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 `Ipopt::InexactNewtonNormalStep::InexactNewtonNormalStep (SmartPtr< AugSystemSolver > aug_solver)`

Default onstructor.

6.59.2.2 `virtual Ipopt::InexactNewtonNormalStep::~~InexactNewtonNormalStep () [virtual]`

Default destructor.

6.59.2.3 `Ipopt::InexactNewtonNormalStep::InexactNewtonNormalStep () [private]`

Default onstructor.

6.59.2.4 `Ipopt::InexactNewtonNormalStep::InexactNewtonNormalStep (const InexactNewtonNormalStep &) [private]`

Copy Constructor.

6.59.3 Member Function Documentation

6.59.3.1 `virtual bool Ipopt::InexactNewtonNormalStep::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

6.59.3.2 `virtual bool Ipopt::InexactNewtonNormalStep::ComputeNewtonNormalStep (Vector & newton_x, Vector & newton_s) [virtual]`

Method for computing the normal step.

The computed step is returned as `normal_x` and `normal_s`, for the `x` and `s` variables, respectively. These quantities are not in the original space, but in the space scaled by the slacks. If the step cannot be computed, this method returns false.

6.59.3.3 `static void Ipopt::InexactNewtonNormalStep::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for IpoptType.

6.59.3.4 `InexactData& Ipopt::InexactNewtonNormalStep::InexData () [inline],[protected]`

Method to easily access Inexact data.

Definition at line 51 of file `IpInexactNewtonNormal.hpp`.

6.59.3.5 `InexactCq& Ipopt::InexactNewtonNormalStep::InexCq () [inline],[protected]`

Method to easily access Inexact calculated quantities.

Definition at line 60 of file `IpInexactNewtonNormal.hpp`.

6.59.3.6 `void Ipopt::InexactNewtonNormalStep::operator= (const InexactNewtonNormalStep &) [private]`

Overloaded Equals Operator.

6.59.4 Member Data Documentation

6.59.4.1 `SmartPtr< AugSystemSolver > Ipopt::InexactNewtonNormalStep::aug_solver_ [private]`

Object to be used to solve the augmented system.

Definition at line 88 of file `IpInexactNewtonNormal.hpp`.

The documentation for this class was generated from the following file:

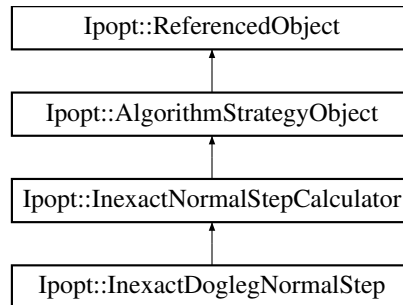
- [Algorithm/Inexact/IpInexactNewtonNormal.hpp](#)

6.60 Ipopt::InexactNormalStepCalculator Class Reference

Base class for computing the normal step for the inexact step calculation algorithm.

```
#include <IpInexactNormalStepCalc.hpp>
```

Inheritance diagram for Ipopt::InexactNormalStepCalculator:



Public Member Functions

- virtual bool `InitializeImpl` (const `OptionsList` &options, const std::string &prefix)=0
overloaded from `AlgorithmStrategyObject`
- virtual bool `ComputeNormalStep` (`SmartPtr`< `Vector` > &normal_x, `SmartPtr`< `Vector` > &normal_s)=0
Method for computing the normal step.

Constructors/Destructors

- `InexactNormalStepCalculator` ()
Default onstructor.
- virtual `~InexactNormalStepCalculator` ()
Default destructor.

Protected Member Functions

- `InexactData` & `InexData` ()
Method to easily access Inexact data.
- `InexactCq` & `InexCq` ()
Method to easily access Inexact calculated quantities.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `InexactNormalStepCalculator` (const `InexactNormalStepCalculator` &)
Copy Constructor.
- void `operator=` (const `InexactNormalStepCalculator` &)
Overloaded Equals Operator.

6.60.1 Detailed Description

Base class for computing the normal step for the inexact step calculation algorithm.

Definition at line 20 of file IpInexactNormalStepCalc.hpp.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 Ipopt::InexactNormalStepCalculator::InexactNormalStepCalculator () [inline]

Default onstructor.

Definition at line 26 of file IpInexactNormalStepCalc.hpp.

6.60.2.2 virtual Ipopt::InexactNormalStepCalculator::~~InexactNormalStepCalculator () [inline],[virtual]

Default destructor.

Definition at line 30 of file IpInexactNormalStepCalc.hpp.

6.60.2.3 Ipopt::InexactNormalStepCalculator::InexactNormalStepCalculator (const InexactNormalStepCalculator &) [private]

Copy Constructor.

6.60.3 Member Function Documentation

6.60.3.1 virtual bool Ipopt::InexactNormalStepCalculator::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::InexactDoglegNormalStep](#).

6.60.3.2 virtual bool Ipopt::InexactNormalStepCalculator::ComputeNormalStep (SmartPtr< Vector > & normal_x, SmartPtr< Vector > & normal_s) [pure virtual]

Method for computing the normal step.

The computed step is returned as normal_x and normal_s, for the x and s variables, respectively. These quantities are not slack-scaled. If the step cannot be computed, this method returns false.

Implemented in [Ipopt::InexactDoglegNormalStep](#).

6.60.3.3 InexactData& Ipopt::InexactNormalStepCalculator::InexData () [inline],[protected]

Method to easily access Inexact data.

Definition at line 47 of file IpInexactNormalStepCalc.hpp.

6.60.3.4 InexactCq& Ipopt::InexactNormalStepCalculator::InexCq () [inline],[protected]

Method to easily access Inexact calculated quantities.

Definition at line 56 of file IpInexactNormalStepCalc.hpp.

6.60.3.5 void Ipopt::InexactNormalStepCalculator::operator=(const InexactNormalStepCalculator &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

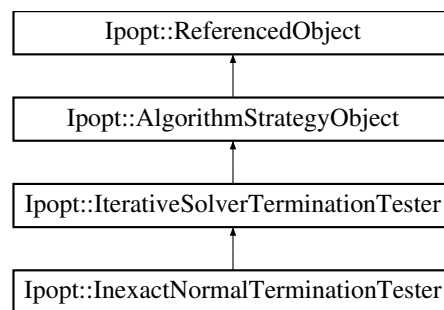
- Algorithm/Inexact/[IpInexactNormalStepCalc.hpp](#)

6.61 Ipopt::InexactNormalTerminationTester Class Reference

This class implements the termination tests for the primal-dual system.

```
#include <IpInexactNormalTerminationTester.hpp>
```

Inheritance diagram for Ipopt::InexactNormalTerminationTester:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
Implementation of the initialization method that has to be overloaded by for each derived class.
- virtual bool [InitializeSolve](#) ()
Method for initializing for the next iterative solve.
- virtual [ETerminationTest](#) [TestTermination](#) ([Index](#) ndim, const [Number](#) *sol, const [Number](#) *resid, [Index](#) iter, [Number](#) norm2_rhs)
This method checks if the current solution of the iterative linear solver is good enough (by returning the corresponding satisfied termination test), or if the Hessian should be modified.
- virtual void [Clear](#) ()
This method can be called after the Solve is over and we can delete anything that has been allocated to free memory.
- virtual [Index](#) [GetSolverIterations](#) () const
Return the number of iterative solver iteration from the most recent solve.
- void [Set_c_Avc_norm_cauchy](#) ([Number](#) c_Avc_norm_cauchy)
Method for setting the normal problem objective function value at the Cauchy step.

/Destructor

- [InexactNormalTerminationTester](#) ()
Default constructor.
- virtual [~InexactNormalTerminationTester](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (SmartPtr< [RegisteredOptions](#) > roptions)

Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [InexactNormalTerminationTester](#) & operator= (const [InexactNormalTerminationTester](#) &)

Overloaded Equals Operator.

Private Attributes

- [Number](#) c_Avc_norm_cauchy_

Value of normal problem objective function achieved by the Cauchy step.

- [Index](#) last_iter_

Last iterative solver iteration counter.

Algorithmic options

- [Number](#) inexact_normal_tol_

Desired reduction of residual.

- [Index](#) inexact_normal_max_iter_

Maximal number of iterative solve iterations.

- bool [requires_scaling_](#)

Is set to true if the linear system is scaled via slacks.

Additional Inherited Members

6.61.1 Detailed Description

This class implements the termination tests for the primal-dual system.

Definition at line 20 of file IpInexactNormalTerminationTester.hpp.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 Ipopt::InexactNormalTerminationTester::InexactNormalTerminationTester ()

Default constructor.

6.61.2.2 virtual Ipopt::InexactNormalTerminationTester::~~InexactNormalTerminationTester () [virtual]

Default destructor.

6.61.3 Member Function Documentation

6.61.3.1 `virtual bool Ipopt::InexactNormalTerminationTester::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

Implementation of the initialization method that has to be overloaded by for each derived class.

Implements [Ipopt::IterativeSolverTerminationTester](#).

6.61.3.2 `static void Ipopt::InexactNormalTerminationTester::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for IpoptType.

6.61.3.3 `virtual bool Ipopt::InexactNormalTerminationTester::InitializeSolve () [virtual]`

Method for initializing for the next iterative solve.

This must be call before the test methods are called.

Implements [Ipopt::IterativeSolverTerminationTester](#).

6.61.3.4 `virtual ETerminationTest Ipopt::InexactNormalTerminationTester::TestTermination (Index ndim, const Number * sol, const Number * resid, Index iter, Number norm2_rhs) [virtual]`

This method checks if the current soltion of the iterative linear solver is good enough (by returning the corresponding satisfied termination test), or if the Hessian should be modified.

The input is the dimension of the augmented system, the current solution vector of the augmented system, the current residual vector.

Implements [Ipopt::IterativeSolverTerminationTester](#).

6.61.3.5 `virtual void Ipopt::InexactNormalTerminationTester::Clear () [virtual]`

This method can be called after the Solve is over and we can delete anything that has been allocated to free memory.

Implements [Ipopt::IterativeSolverTerminationTester](#).

6.61.3.6 `virtual Index Ipopt::InexactNormalTerminationTester::GetSolverIterations () const [inline],[virtual]`

Return the number of iterative solver iteration from the most recent solve.

Implements [Ipopt::IterativeSolverTerminationTester](#).

Definition at line 63 of file `IpInexactNormalTerminationTester.hpp`.

6.61.3.7 `void Ipopt::InexactNormalTerminationTester::Set_c_Avc_norm_cauchy (Number c_Avc_norm_cauchy) [inline]`

Method for setting the normal problem objective function value at the Cauchy step.

This must be called by the Dogleg object.

Definition at line 71 of file `IpInexactNormalTerminationTester.hpp`.

6.61.3.8 `InexactNormalTerminationTester& Ipopt::InexactNormalTerminationTester::operator= (const InexactNormalTerminationTester &) [private]`

Overloaded Equals Operator.

6.61.4 Member Data Documentation

6.61.4.1 Number Ipopt::InexactNormalTerminationTester::inexact_normal_tol_ [private]

Desired reduction of residual.

Definition at line 92 of file IpInexactNormalTerminationTester.hpp.

6.61.4.2 Index Ipopt::InexactNormalTerminationTester::inexact_normal_max_iter_ [private]

Maximal number of iterative solve iterations.

Definition at line 94 of file IpInexactNormalTerminationTester.hpp.

6.61.4.3 bool Ipopt::InexactNormalTerminationTester::requires_scaling_ [private]

Is set to true if the linear system is scaled via slacks.

Definition at line 96 of file IpInexactNormalTerminationTester.hpp.

6.61.4.4 Number Ipopt::InexactNormalTerminationTester::c_Avc_norm_cauchy_ [private]

Value of normal problem objective function achieved by the Cauchy step.

This must be set by the Dogleg step object.

Definition at line 101 of file IpInexactNormalTerminationTester.hpp.

6.61.4.5 Index Ipopt::InexactNormalTerminationTester::last_iter_ [private]

Last iterative solver iteration counter.

Definition at line 104 of file IpInexactNormalTerminationTester.hpp.

The documentation for this class was generated from the following file:

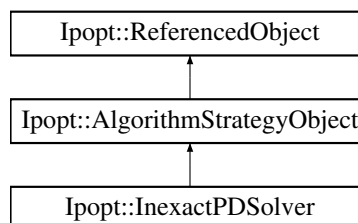
- Algorithm/Inexact/[IpInexactNormalTerminationTester.hpp](#)

6.62 Ipopt::InexactPDSolver Class Reference

This is the implemetation of the Primal-Dual System, allowing the usage of an inexact linear solver.

```
#include <IpInexactPDSolver.hpp>
```

Inheritance diagram for Ipopt::InexactPDSolver:



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)

Implementation of the initialization method that has to be overloaded by for each derived class.

- virtual bool **Solve** (const **IteratesVector** &rhs, **IteratesVector** &sol)

Solve the primal dual system, given one right hand side.

/Destructor

- **InexactPDSolver** (**AugSystemSolver** &augSysSolver, **PDPerturbationHandler** &perturbHandler)

Constructor that takes in the Augmented System solver that is to be used inside.

- virtual **~InexactPDSolver** ()

Default destructor.

Static Public Member Functions

- static void **RegisterOptions** (**SmartPtr**< **RegisteredOptions** > roptions)

Methods for IpoptType.

Private Member Functions

- **InexactData** & **InexData** ()

Method to easily access Inexact data.

- **InexactCq** & **InexCq** ()

Method to easily access Inexact calculated quantities.

- void **ComputeResiduals** (const **SymMatrix** &W, const **Matrix** &J_c, const **Matrix** &J_d, const **Matrix** &Pd_L, const **Matrix** &Pd_U, const **Vector** &v_L, const **Vector** &v_U, const **Vector** &slack_s_L, const **Vector** &slack_s_U, const **Vector** &sigma_s, const **IteratesVector** &rhs, const **IteratesVector** &res, **IteratesVector** &resid)

Internal function for computing the residual (resid) given the right hand side (rhs) and the solution of the system (res).

- bool **HessianRequiresChange** ()

Method for checking if the Hessian matrix has to be modified.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- **InexactPDSolver** ()

Default Constructor.

- **InexactPDSolver** & **operator=** (const **InexactPDSolver** &)

Overloaded Equals Operator.

Private Attributes

- bool **is_pardiso_**

flag indicating if we are dealing with the Pardiso solver (temporary)

- **Index last_info_ls_count_**

Strategy objects to hold on to.

- **SmartPtr**< **AugSystemSolver** > **augSysSolver_**

Pointer to the Solver for the augmented system.

- **SmartPtr**< **PDPerturbationHandler** > **perturbHandler_**

Pointer to the Perturbation Handler.

Algorithmic options

- [Number tcc_psi_](#)
Psi factor in the tangential component condition.
- [Number tcc_theta_](#)
theta factor in the tangential component condition
- [Number tcc_theta_mu_exponent_](#)
mu exponent when multiplied to theta in the tangential component condition
- [bool modify_hessian_with_slacks_](#)
flag indicating if the Hessian for the (s,s) part should be modified with the slacks instead of the identity matrix
- [Index inexact_regularization_ls_count_trigger_](#)
Threshold on line search evaluation count to trigger Hessia modification.

Additional Inherited Members

6.62.1 Detailed Description

This is the implemetation of the Primal-Dual System, allowing the usage of an inexact linear solver.

The step computed is usually for the tangential step.

Definition at line 24 of file `IpInexactPDSolver.hpp`.

6.62.2 Constructor & Destructor Documentation

6.62.2.1 `Ipopt::InexactPDSolver::InexactPDSolver (AugSystemSolver & augSysSolver, PDPerturbationHandler & perturbHandler)`

Constructor that takes in the Augmented System solver that is to be used inside.

6.62.2.2 `virtual Ipopt::InexactPDSolver::~~InexactPDSolver () [virtual]`

Default destructor.

6.62.2.3 `Ipopt::InexactPDSolver::InexactPDSolver () [private]`

Default Constructor.

6.62.3 Member Function Documentation

6.62.3.1 `bool Ipopt::InexactPDSolver::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

Implementation of the initialization method that has to be overloaded by for each derived class.

Implements [Ipopt::AlgorithmStrategyObject](#).

6.62.3.2 `virtual bool Ipopt::InexactPDSolver::Solve (const IteratesVector & rhs, IteratesVector & sol) [virtual]`

Solve the primal dual system, given one right hand side.

6.62.3.3 `static void Ipopt::InexactPDSolver::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for IpoptType.

6.62.3.4 InexactPDSolver& Ipopt::InexactPDSolver::operator= (const InexactPDSolver &) [private]

Overloaded Equals Operator.

6.62.3.5 InexactData& Ipopt::InexactPDSolver::InexData () [inline],[private]

Method to easily access Inexact data.

Definition at line 69 of file IpInexactPDSolver.hpp.

6.62.3.6 InexactCq& Ipopt::InexactPDSolver::InexCq () [inline],[private]

Method to easily access Inexact calculated quantities.

Definition at line 78 of file IpInexactPDSolver.hpp.

6.62.3.7 void Ipopt::InexactPDSolver::ComputeResiduals (const SymMatrix & W, const Matrix & J_c, const Matrix & J_d, const Matrix & Pd_L, const Matrix & Pd_U, const Vector & v_L, const Vector & v_U, const Vector & slack_s_L, const Vector & slack_s_U, const Vector & sigma_s, const IteratesVector & rhs, const IteratesVector & res, IteratesVector & resid) [private]

Internal function for computing the residual (resid) given the right hand side (rhs) and the solution of the system (res).

6.62.3.8 bool Ipopt::InexactPDSolver::HessianRequiresChange () [private]

Method for checking if the Hessian matrix has to be modified.

All required data is obtained from the Data objects, so those values have to be set before this is called.

6.62.4 Member Data Documentation

6.62.4.1 SmartPtr<AugSystemSolver> Ipopt::InexactPDSolver::augSysSolver_ [private]

Pointer to the Solver for the augmented system.

Definition at line 89 of file IpInexactPDSolver.hpp.

6.62.4.2 SmartPtr<PDPerturbationHandler> Ipopt::InexactPDSolver::perturbHandler_ [private]

Pointer to the Perturbation Handler.

Definition at line 91 of file IpInexactPDSolver.hpp.

6.62.4.3 Number Ipopt::InexactPDSolver::tcc_psi_ [private]

Psi factor in the tangential component condition.

Definition at line 119 of file IpInexactPDSolver.hpp.

6.62.4.4 Number Ipopt::InexactPDSolver::tcc_theta_ [private]

theta factor in the tangential component condition

Definition at line 121 of file IpInexactPDSolver.hpp.

6.62.4.5 Number Ipopt::InexactPDSolver::tcc_theta_mu_exponent_ [private]

mu exponent when multiplied to theta in the tangential component condition

Definition at line 124 of file IpInexactPDSolver.hpp.

6.62.4.6 `bool Ipopt::InexactPDSolver::modify_hessian_with_slacks_ [private]`

flag indicating if the Hessian for the (s,s) part should be modified with the slacks instead of the identity matrix

Definition at line 127 of file `IpInexactPDSolver.hpp`.

6.62.4.7 `Index Ipopt::InexactPDSolver::inexact_regularization_ls_count_trigger_ [private]`

Threshold on line search evaluation count to trigger Hessia modification.

Definition at line 130 of file `IpInexactPDSolver.hpp`.

6.62.4.8 `bool Ipopt::InexactPDSolver::is_pardiso_ [private]`

flag indicating if we are dealing with the Pardiso solver (temporary)

Definition at line 135 of file `IpInexactPDSolver.hpp`.

6.62.4.9 `Index Ipopt::InexactPDSolver::last_info_ls_count_ [private]`

Definition at line 137 of file `IpInexactPDSolver.hpp`.

The documentation for this class was generated from the following file:

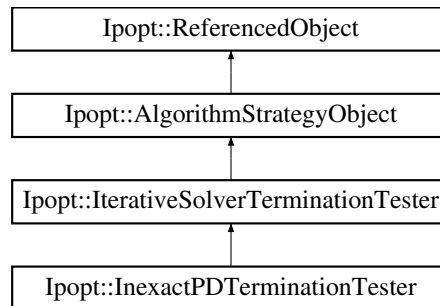
- Algorithm/Inexact/[IpInexactPDSolver.hpp](#)

6.63 Ipopt::InexactPDTerminationTester Class Reference

This class implements the termination tests for the primal-dual system.

```
#include <IpInexactPDTerminationTester.hpp>
```

Inheritance diagram for `Ipopt::InexactPDTerminationTester`:



Public Member Functions

- virtual `bool InitializeImpl` (const [OptionsList](#) &options, const `std::string` &prefix)
Implementation of the initialization method that has to be overloaded by for each derived class.
- virtual `bool InitializeSolve` ()
Method for initializing for the next iterative solve.
- virtual `ETerminationTest TestTermination` (`Index` ndim, const `Number` *sol, const `Number` *resid, `Index` iter, `Number` norm2_rhs)
This method checks if the current solution of the iterative linear solver is good enough (by returning the corresponding satisfied termination test), or if the Hessian should be modified.

- virtual void [Clear](#) ()

This method can be called after the Solve is over and we can delete anything that has been allocated to free memory.

- virtual [Index GetSolverIterations](#) () const

Return the number of iterative solver iteration from the most recent solve.

/Destructor

- [InexactPDTerminationTester](#) ()

Default constructor.

- virtual [~InexactPDTerminationTester](#) ()

Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)

Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [InexactPDTerminationTester](#) & [operator=](#) (const [InexactPDTerminationTester](#) &)

Overloaded Equals Operator.

Private Attributes

- [Index last_iter_](#)

Last iterative solver iteration counter.

Algorithmic options

- [Number tcc_psi_](#)

Psi factor in the tangential component condition.

- [Number tcc_theta_](#)

theta factor in the tangential component condition

- [Number tcc_theta_mu_exponent_](#)

mu exponent when multiplied to theta in the tangential component condition

- [Number tcc_zeta_](#)

zeta factor in the tangential component condition

- [Number tt_kappa1_](#)

kappa_1 factor in termination test 1

- [Number tt_kappa2_](#)

kappa_2 factor in termination test 2

- [Number tt_eps2_](#)

eps_2 constant in termination test 2

- [Number tt_eps3_](#)

eps_3 constant in termination test 3

- [Number rho_](#)

- *rho* constant from penalty parameter update.
- Number `inexact_desired_pd_residual_`
Desired reduction of residual.
- Index `inexact_desired_pd_residual_iter_`
Number of iterations allowed for desired pd residual.
- bool `requires_scaling_`
Is set to true if the linear system is scaled via slacks.

Quantities that are identical for all tests and can be

precomputed

- SmartPtr< const Vector > `curr_Av_c_`
- SmartPtr< const Vector > `curr_Av_d_`
- Number `c_norm_`
- Number `c_plus_Av_norm_`
- Number `v_norm_scaled_`
- SmartPtr< const Vector > `curr_grad_barrier_obj_x_`
- SmartPtr< const Vector > `curr_grad_barrier_obj_s_`
- SmartPtr< const Matrix > `curr_jac_c_`
- SmartPtr< const Matrix > `curr_jac_d_`
- SmartPtr< const Vector > `curr_scaling_slacks_`
- SmartPtr< Vector > `curr_nabla_phi_plus_ATy_x_`
- SmartPtr< Vector > `curr_nabla_phi_plus_ATy_s_`
- Number `curr_Av_norm_`
- Number `curr_tt1_norm_`
- Number `curr_tt2_norm_`
- SmartPtr< const Vector > `curr_Wv_x_`
- SmartPtr< const Vector > `curr_Wv_s_`
- bool `try_tt2_`

Quantities from previous iteration required in the

tests

- Number `last_Av_norm_`
- Number `last_tt1_norm_`

Additional Inherited Members

6.63.1 Detailed Description

This class implements the termination tests for the primal-dual system.

Definition at line 20 of file `IpInexactPDTerminationTester.hpp`.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 Ipopt::InexactPDTerminationTester::InexactPDTerminationTester ()

Default constructor.

6.63.2.2 virtual Ipopt::InexactPDTerminationTester::~InexactPDTerminationTester () [virtual]

Default destructor.

6.63.3 Member Function Documentation

6.63.3.1 `virtual bool Ipopt::InexactPDTerminationTester::InitializeImpl (const OptionsList & options, const std::string & prefix)`
[virtual]

Implementation of the initialization method that has to be overloaded by for each derived class.

Implements [Ipopt::IterativeSolverTerminationTester](#).

6.63.3.2 `static void Ipopt::InexactPDTerminationTester::RegisterOptions (SmartPtr< RegisteredOptions > roptions)`
[static]

Methods for IpoptType.

6.63.3.3 `virtual bool Ipopt::InexactPDTerminationTester::InitializeSolve ()` [virtual]

Method for initializing for the next iterative solve.

This must be call before the test methods are called.

Implements [Ipopt::IterativeSolverTerminationTester](#).

6.63.3.4 `virtual ETerminationTest Ipopt::InexactPDTerminationTester::TestTermination (Index ndim, const Number * sol, const Number * resid, Index iter, Number norm2_rhs)` [virtual]

This method checks if the current soltion of the iterative linear solver is good enough (by returning the corresponding satisfied termination test), or if the Hessian should be modified.

The input is the dimension of the augmented system, the current solution vector of the augmented system, the current residual vector.

Implements [Ipopt::IterativeSolverTerminationTester](#).

6.63.3.5 `virtual void Ipopt::InexactPDTerminationTester::Clear ()` [virtual]

This method can be called after the Solve is over and we can delete anything that has been allocated to free memory.

Implements [Ipopt::IterativeSolverTerminationTester](#).

6.63.3.6 `virtual Index Ipopt::InexactPDTerminationTester::GetSolverIterations () const` [inline],[virtual]

Return the number of iterative solver iteration from the most recent solve.

Implements [Ipopt::IterativeSolverTerminationTester](#).

Definition at line 62 of file `IpInexactPDTerminationTester.hpp`.

6.63.3.7 `InexactPDTerminationTester& Ipopt::InexactPDTerminationTester::operator= (const InexactPDTerminationTester &)` [private]

Overloaded Equals Operator.

6.63.4 Member Data Documentation

6.63.4.1 `Number Ipopt::InexactPDTerminationTester::tcc_psi_` [private]

Psi factor in the tangential component condition.

Definition at line 83 of file `IpInexactPDTerminationTester.hpp`.

6.63.4.2 Number Ipopt::InexactPDTerminationTester::tcc_theta_ [private]

theta factor in the tangential component condition

Definition at line 85 of file IpInexactPDTerminationTester.hpp.

6.63.4.3 Number Ipopt::InexactPDTerminationTester::tcc_theta_mu_exponent_ [private]

mu exponent when multiplied to theta in the tangential component condition

Definition at line 88 of file IpInexactPDTerminationTester.hpp.

6.63.4.4 Number Ipopt::InexactPDTerminationTester::tcc_zeta_ [private]

zeta factor in the tangential component condition

Definition at line 90 of file IpInexactPDTerminationTester.hpp.

6.63.4.5 Number Ipopt::InexactPDTerminationTester::tt_kappa1_ [private]

kappa_1 factor in termination test 1

Definition at line 92 of file IpInexactPDTerminationTester.hpp.

6.63.4.6 Number Ipopt::InexactPDTerminationTester::tt_kappa2_ [private]

kappa_2 factor in termination test 2

Definition at line 94 of file IpInexactPDTerminationTester.hpp.

6.63.4.7 Number Ipopt::InexactPDTerminationTester::tt_eps2_ [private]

eps_2 constant in termination test 2

Definition at line 96 of file IpInexactPDTerminationTester.hpp.

6.63.4.8 Number Ipopt::InexactPDTerminationTester::tt_eps3_ [private]

eps_3 constant in termination test 3

Definition at line 98 of file IpInexactPDTerminationTester.hpp.

6.63.4.9 Number Ipopt::InexactPDTerminationTester::rho_ [private]

rho constant from penalty parameter update.

This is called τ_π in MIPS paper

Definition at line 101 of file IpInexactPDTerminationTester.hpp.

6.63.4.10 Number Ipopt::InexactPDTerminationTester::inexact_desired_pd_residual_ [private]

Desired reduction of residual.

Definition at line 103 of file IpInexactPDTerminationTester.hpp.

6.63.4.11 Index Ipopt::InexactPDTerminationTester::inexact_desired_pd_residual_iter_ [private]

Number of iterations allowed for desired pd residual.

Definition at line 105 of file IpInexactPDTerminationTester.hpp.

6.63.4.12 **bool** Ipopt::InexactPDTerminationTester::requires_scaling_ [private]

Is set to true if the linear system is scaled via slacks.

Definition at line 107 of file IpInexactPDTerminationTester.hpp.

6.63.4.13 **SmartPtr<const Vector>** Ipopt::InexactPDTerminationTester::curr_Av_c_ [private]

Definition at line 113 of file IpInexactPDTerminationTester.hpp.

6.63.4.14 **SmartPtr<const Vector>** Ipopt::InexactPDTerminationTester::curr_Av_d_ [private]

Definition at line 114 of file IpInexactPDTerminationTester.hpp.

6.63.4.15 **Number** Ipopt::InexactPDTerminationTester::c_norm_ [private]

Definition at line 115 of file IpInexactPDTerminationTester.hpp.

6.63.4.16 **Number** Ipopt::InexactPDTerminationTester::c_plus_Av_norm_ [private]

Definition at line 116 of file IpInexactPDTerminationTester.hpp.

6.63.4.17 **Number** Ipopt::InexactPDTerminationTester::v_norm_scaled_ [private]

Definition at line 117 of file IpInexactPDTerminationTester.hpp.

6.63.4.18 **SmartPtr<const Vector>** Ipopt::InexactPDTerminationTester::curr_grad_barrier_obj_x_ [private]

Definition at line 118 of file IpInexactPDTerminationTester.hpp.

6.63.4.19 **SmartPtr<const Vector>** Ipopt::InexactPDTerminationTester::curr_grad_barrier_obj_s_ [private]

Definition at line 119 of file IpInexactPDTerminationTester.hpp.

6.63.4.20 **SmartPtr<const Matrix>** Ipopt::InexactPDTerminationTester::curr_jac_c_ [private]

Definition at line 120 of file IpInexactPDTerminationTester.hpp.

6.63.4.21 **SmartPtr<const Matrix>** Ipopt::InexactPDTerminationTester::curr_jac_d_ [private]

Definition at line 121 of file IpInexactPDTerminationTester.hpp.

6.63.4.22 **SmartPtr<const Vector>** Ipopt::InexactPDTerminationTester::curr_scaling_slacks_ [private]

Definition at line 122 of file IpInexactPDTerminationTester.hpp.

6.63.4.23 **SmartPtr<Vector>** Ipopt::InexactPDTerminationTester::curr_nabla_phi_plus_ATy_x_ [private]

Definition at line 123 of file IpInexactPDTerminationTester.hpp.

6.63.4.24 **SmartPtr<Vector>** Ipopt::InexactPDTerminationTester::curr_nabla_phi_plus_ATy_s_ [private]

Definition at line 124 of file IpInexactPDTerminationTester.hpp.

6.63.4.25 **Number** Ipopt::InexactPDTerminationTester::curr_Av_norm_ [private]

Definition at line 125 of file IpInexactPDTerminationTester.hpp.

6.63.4.26 **Number** Ipopt::InexactPDTerminationTester::curr_tt1_norm_ [private]

Definition at line 126 of file IpInexactPDTerminationTester.hpp.

6.63.4.27 **Number** Ipopt::InexactPDTerminationTester::curr_tt2_norm_ [private]

Definition at line 127 of file IpInexactPDTerminationTester.hpp.

6.63.4.28 **SmartPtr<const Vector>** Ipopt::InexactPDTerminationTester::curr_Wv_x_ [private]

Definition at line 128 of file IpInexactPDTerminationTester.hpp.

6.63.4.29 **SmartPtr<const Vector>** Ipopt::InexactPDTerminationTester::curr_Wv_s_ [private]

Definition at line 129 of file IpInexactPDTerminationTester.hpp.

6.63.4.30 **bool** Ipopt::InexactPDTerminationTester::try_tt2_ [private]

Definition at line 130 of file IpInexactPDTerminationTester.hpp.

6.63.4.31 **Number** Ipopt::InexactPDTerminationTester::last_Av_norm_ [private]

Definition at line 136 of file IpInexactPDTerminationTester.hpp.

6.63.4.32 **Number** Ipopt::InexactPDTerminationTester::last_tt1_norm_ [private]

Definition at line 137 of file IpInexactPDTerminationTester.hpp.

6.63.4.33 **Index** Ipopt::InexactPDTerminationTester::last_iter_ [private]

Last iterative solver iteration counter.

Definition at line 141 of file IpInexactPDTerminationTester.hpp.

The documentation for this class was generated from the following file:

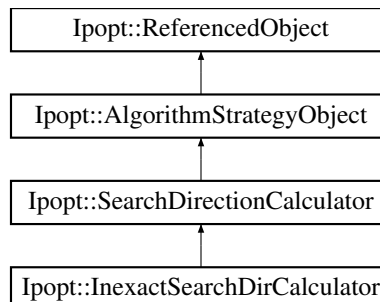
- Algorithm/Inexact/[IpInexactPDTerminationTester.hpp](#)

6.64 Ipopt::InexactSearchDirCalculator Class Reference

Implementation of the search direction calculator that computes the search direction using iterative linear solvers.

```
#include <IpInexactSearchDirCalc.hpp>
```

Inheritance diagram for Ipopt::InexactSearchDirCalculator:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [ComputeSearchDirection](#) ()
Method for computing the search direction.

Constructors/Destructors

- [InexactSearchDirCalculator](#) ([SmartPtr](#)< [InexactNormalStepCalculator](#) > normal_step_calculator, [SmartPtr](#)< [InexactPDSolver](#) > inexact_pd_solver)
Constructor.
- virtual [~InexactSearchDirCalculator](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Types

- enum [DecompositionTypeEnum](#) { [ALWAYS](#) =0, [ADAPTIVE](#), [SWITCH_ONCE](#) }
enumeration for decomposition options

Private Member Functions

- [InexactData](#) & [InexData](#) ()
Method to easily access Inexact data.
- [InexactCq](#) & [InexCq](#) ()
Method to easily access Inexact calculated quantities.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [InexactSearchDirCalculator](#) ()
Default Constructor.
- [InexactSearchDirCalculator](#) (const [InexactSearchDirCalculator](#) &)
Copy Constructor.
- void [operator=](#) (const [InexactSearchDirCalculator](#) &)
Overloaded Equals Operator.

Private Attributes

- [DecompositionTypeEnum decomposition_type_](#)
Type of decomposition.

Algorithmic options

- [Number local_inf_Ac_tol_](#)
termination tolerance for local infeasibility

Strategy objects

- [SmartPtr](#)
< [InexactNormalStepCalculator](#) > [normal_step_calculator_](#)
- [SmartPtr](#)< [InexactPDSolver](#) > [inexact_pd_solver_](#)

Additional Inherited Members

6.64.1 Detailed Description

Implementation of the search direction calculator that computes the search direction using iterative linear solvers.

Those steps do not necessarily satisfy the linearized KKT conditions with high accuracy.

Definition at line 24 of file IpInexactSearchDirCalc.hpp.

6.64.2 Member Enumeration Documentation

6.64.2.1 `enum Ipopt::InexactSearchDirCalculator::DecompositionTypeEnum [private]`

enumeration for decomposition options

Enumerator

ALWAYS
ADAPTIVE
SWITCH_ONCE

Definition at line 102 of file IpInexactSearchDirCalc.hpp.

6.64.3 Constructor & Destructor Documentation

6.64.3.1 `Ipopt::InexactSearchDirCalculator::InexactSearchDirCalculator (SmartPtr< InexactNormalStepCalculator > normal_step_calculator, SmartPtr< InexactPDSolver > inexact_pd_solver)`

Constructor.

6.64.3.2 `virtual Ipopt::InexactSearchDirCalculator::~~InexactSearchDirCalculator () [virtual]`

Default destructor.

6.64.3.3 `Ipopt::InexactSearchDirCalculator::InexactSearchDirCalculator () [private]`

Default Constructor.

6.64.3.4 `Ipopt::InexactSearchDirCalculator::InexactSearchDirCalculator (const InexactSearchDirCalculator &)`
`[private]`

Copy Constructor.

6.64.4 Member Function Documentation

6.64.4.1 `virtual bool Ipopt::InexactSearchDirCalculator::InitializeImpl (const OptionsList & options, const std::string & prefix)`
`[virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SearchDirectionCalculator](#).

6.64.4.2 `virtual bool Ipopt::InexactSearchDirCalculator::ComputeSearchDirection ()` `[virtual]`

Method for computing the search direction.

In this version, we compute a normal and a tangential component, which are stored in the [InexactData](#) object. The overall step is still stored in the [IpoptData](#) object.

Implements [Ipopt::SearchDirectionCalculator](#).

6.64.4.3 `static void Ipopt::InexactSearchDirCalculator::RegisterOptions (SmartPtr< RegisteredOptions > roptions)`
`[static]`

Methods for IpoptType.

6.64.4.4 `void Ipopt::InexactSearchDirCalculator::operator= (const InexactSearchDirCalculator &)` `[private]`

Overloaded Equals Operator.

6.64.4.5 `InexactData& Ipopt::InexactSearchDirCalculator::InexData ()` `[inline],[private]`

Method to easily access Inexact data.

Definition at line 72 of file `IpInexactSearchDirCalc.hpp`.

6.64.4.6 `InexactCq& Ipopt::InexactSearchDirCalculator::InexCq ()` `[inline],[private]`

Method to easily access Inexact calculated quantities.

Definition at line 81 of file `IpInexactSearchDirCalc.hpp`.

6.64.5 Member Data Documentation

6.64.5.1 `Number Ipopt::InexactSearchDirCalculator::local_inf_Ac_tol_` `[private]`

termination tolerance for local infeasibility

Definition at line 92 of file `IpInexactSearchDirCalc.hpp`.

6.64.5.2 `SmartPtr<InexactNormalStepCalculator> Ipopt::InexactSearchDirCalculator::normal_step_calculator_`
`[private]`

Definition at line 97 of file `IpInexactSearchDirCalc.hpp`.

6.64.5.3 **SmartPtr<InexactPDSolver>** Ipopt::InexactSearchDirCalculator::inexact_pd_solver_ [private]

Definition at line 98 of file IpInexactSearchDirCalc.hpp.

6.64.5.4 **DecompositionTypeEnum** Ipopt::InexactSearchDirCalculator::decomposition_type_ [private]

Type of decomposition.

Definition at line 109 of file IpInexactSearchDirCalc.hpp.

The documentation for this class was generated from the following file:

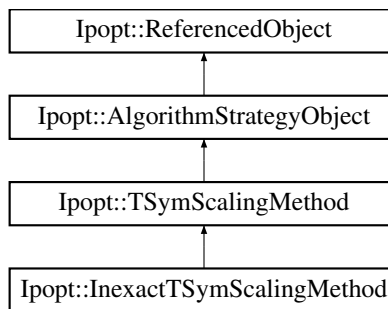
- Algorithm/Inexact/[IpInexactSearchDirCalc.hpp](#)

6.65 Ipopt::InexactTSymScalingMethod Class Reference

Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm.

```
#include <IpInexactTSymScalingMethod.hpp>
```

Inheritance diagram for Ipopt::InexactTSymScalingMethod:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [ComputeSymTScalingFactors](#) ([Index](#) n, [Index](#) nnz, const [ipfint](#) *airn, const [ipfint](#) *ajcn, const double *a, double *scaling_factors)
Method for computing the symmetric scaling factors, given the symmetric matrix in triplet (MA27) format.

Constructor/Destructor

- [InexactTSymScalingMethod](#) ()
- virtual [~InexactTSymScalingMethod](#) ()

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [InexactTSymScalingMethod](#) (const [InexactTSymScalingMethod](#) &)
Copy Constructor.
- void [operator=](#) (const [InexactTSymScalingMethod](#) &)
Overloaded Equals Operator.
- [InexactCq](#) & [InexCq](#) ()
Method to easily access Inexact calculated quantities.

Additional Inherited Members

6.65.1 Detailed Description

Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm.

The scaling is only considering the current slacks.

Definition at line 24 of file `IpInexactTSymScalingMethod.hpp`.

6.65.2 Constructor & Destructor Documentation

6.65.2.1 `Ipopt::InexactTSymScalingMethod::InexactTSymScalingMethod ()` `[inline]`

Definition at line 29 of file `IpInexactTSymScalingMethod.hpp`.

6.65.2.2 `virtual Ipopt::InexactTSymScalingMethod::~~InexactTSymScalingMethod ()` `[inline]`, `[virtual]`

Definition at line 32 of file `IpInexactTSymScalingMethod.hpp`.

6.65.2.3 `Ipopt::InexactTSymScalingMethod::InexactTSymScalingMethod (const InexactTSymScalingMethod &)` `[private]`

Copy Constructor.

6.65.3 Member Function Documentation

6.65.3.1 `virtual bool Ipopt::InexactTSymScalingMethod::InitializImpl (const OptionsList & options, const std::string & prefix)` `[virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::TSymScalingMethod](#).

6.65.3.2 `virtual bool Ipopt::InexactTSymScalingMethod::ComputeSymTScalingFactors (Index n, Index nnz, const ipfint * airn, const ipfint * ajcn, const double * a, double * scaling_factors)` `[virtual]`

Method for computing the symmetric scaling factors, given the symmetric matrix in triplet (MA27) format.

6.65.3.3 `void Ipopt::InexactTSymScalingMethod::operator= (const InexactTSymScalingMethod &)` `[private]`

Overloaded Equals Operator.

6.65.3.4 `InexactCq & Ipopt::InexactTSymScalingMethod::InexCq ()` `[inline]`, `[private]`

Method to easily access Inexact calculated quantities.

Definition at line 62 of file `IpInexactTSymScalingMethod.hpp`.

The documentation for this class was generated from the following file:

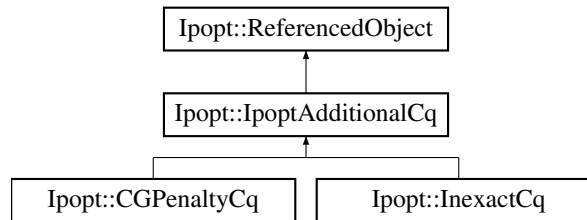
- [Algorithm/Inexact/IpInexactTSymScalingMethod.hpp](#)

6.66 Ipopt::IpoptAdditionalCq Class Reference

Base class for additional calculated quantities that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers.

```
#include <IpIpoptCalculatedQuantities.hpp>
```

Inheritance diagram for Ipopt::IpoptAdditionalCq:



Public Member Functions

- virtual bool [Initialize](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)=0
This method is called to initialize the global algorithmic parameters.

Constructors/Destructors

- [IpoptAdditionalCq](#) ()
Default Constructor.
- virtual [~IpoptAdditionalCq](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IpoptAdditionalCq](#) (const [IpoptAdditionalCq](#) &)
Copy Constructor.
- void [operator=](#) (const [IpoptAdditionalCq](#) &)
Overloaded Equals Operator.

6.66.1 Detailed Description

Base class for additional calculated quantities that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers.

The regular [IpoptCalculatedQuantities](#) object should be given a derivation of this base class when it is created.

Definition at line 40 of file [IpIpoptCalculatedQuantities.hpp](#).

6.66.2 Constructor & Destructor Documentation

6.66.2.1 `Ipopt::IpoptAdditionalCq::IpoptAdditionalCq ()` `[inline]`

Default Constructor.

Definition at line 46 of file `IpIpoptCalculatedQuantities.hpp`.

6.66.2.2 `virtual Ipopt::IpoptAdditionalCq::~~IpoptAdditionalCq ()` `[inline],[virtual]`

Default destructor.

Definition at line 50 of file `IpIpoptCalculatedQuantities.hpp`.

6.66.2.3 `Ipopt::IpoptAdditionalCq::IpoptAdditionalCq (const IpoptAdditionalCq &)` `[private]`

Copy Constructor.

6.66.3 Member Function Documentation

6.66.3.1 `virtual bool Ipopt::IpoptAdditionalCq::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix)` `[pure virtual]`

This method is called to initialize the global algorithmic parameters.

The parameters are taken from the [OptionsList](#) object.

Implemented in [Ipopt::InexactCq](#), and [Ipopt::CGPenaltyCq](#).

6.66.3.2 `void Ipopt::IpoptAdditionalCq::operator= (const IpoptAdditionalCq &)` `[private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

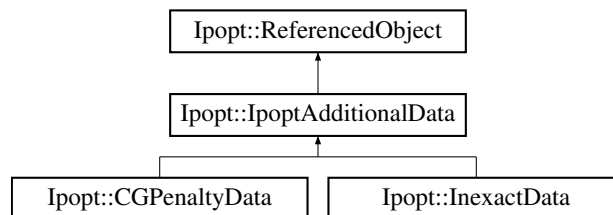
- [Algorithm/IpIpoptCalculatedQuantities.hpp](#)

6.67 Ipopt::IpoptAdditionalData Class Reference

Base class for additional data that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers.

```
#include <IpIpoptData.hpp>
```

Inheritance diagram for `Ipopt::IpoptAdditionalData`:



Public Member Functions

- virtual bool [Initialize](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)=0
This method is called to initialize the global algorithmic parameters.
- virtual bool [InitializeDataStructures](#) ()=0
Initialize Data Structures at the beginning.
- virtual void [AcceptTrialPoint](#) ()=0
Do whatever is necessary to accept a trial point as current iterate.

Constructors/Destructors

- [IpoptAdditionalData](#) ()
Default Constructor.
- virtual [~IpoptAdditionalData](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IpoptAdditionalData](#) (const [IpoptAdditionalData](#) &)
Copy Constructor.
- void [operator=](#) (const [IpoptAdditionalData](#) &)
Overloaded Equals Operator.

6.67.1 Detailed Description

Base class for additional data that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers.

The regular [IpoptData](#) object should be given a derivation of this base class when it is created.

Definition at line 28 of file `IpIpoptData.hpp`.

6.67.2 Constructor & Destructor Documentation

6.67.2.1 Ipopt::IpoptAdditionalData::IpoptAdditionalData () [inline]

Default Constructor.

Definition at line 34 of file `IpIpoptData.hpp`.

6.67.2.2 virtual Ipopt::IpoptAdditionalData::~~IpoptAdditionalData () [inline], [virtual]

Default destructor.

Definition at line 38 of file `IpIpoptData.hpp`.

6.67.2.3 Ipopt::IpoptAdditionalData::IpoptAdditionalData (const IpoptAdditionalData &) [private]

Copy Constructor.

6.67.3 Member Function Documentation

6.67.3.1 `virtual bool Ipopt::IpoptAdditionalData::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [pure virtual]`

This method is called to initialize the global algorithmic parameters.

The parameters are taken from the [OptionsList](#) object.

Implemented in [Ipopt::CGPenaltyData](#), and [Ipopt::InexactData](#).

6.67.3.2 `virtual bool Ipopt::IpoptAdditionalData::InitializeDataStructures () [pure virtual]`

Initialize Data Structures at the beginning.

Implemented in [Ipopt::CGPenaltyData](#), and [Ipopt::InexactData](#).

6.67.3.3 `virtual void Ipopt::IpoptAdditionalData::AcceptTrialPoint () [pure virtual]`

Do whatever is necessary to accept a trial point as current iterate.

This is also used to finish an iteration, i.e., to release memory, and to reset any flags for a new iteration.

Implemented in [Ipopt::CGPenaltyData](#), and [Ipopt::InexactData](#).

6.67.3.4 `void Ipopt::IpoptAdditionalData::operator= (const IpoptAdditionalData &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

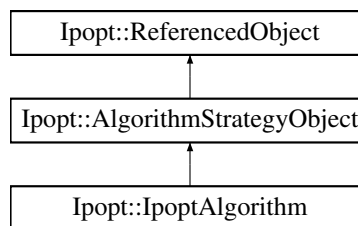
- [Algorithm/IpIpoptData.hpp](#)

6.68 Ipopt::IpoptAlgorithm Class Reference

The main ipopt algorithm class.

```
#include <IpIpoptAlg.hpp>
```

Inheritance diagram for Ipopt::IpoptAlgorithm:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- [SolverReturn Optimize](#) (bool isResto=false)
Main solve method.

Constructors/Destructors

- [IpoptAlgorithm](#) (const [SmartPtr](#)< [SearchDirectionCalculator](#) > &search_dir_calculator, const [SmartPtr](#)< [LineSearch](#) > &line_search, const [SmartPtr](#)< [MuUpdate](#) > &mu_update, const [SmartPtr](#)< [ConvergenceCheck](#) > &conv_check, const [SmartPtr](#)< [IterateInitializer](#) > &iterate_initializer, const [SmartPtr](#)< [IterationOutput](#) > &iter_output, const [SmartPtr](#)< [HessianUpdater](#) > &hessian_updater, const [SmartPtr](#)< [EqMultiplierCalculator](#) > &eq_multiplier_calculator=NULL)
Constructor.
- virtual [~IpoptAlgorithm](#) ()
Default destructor.

Access to internal strategy objects

- [SmartPtr](#)
< [SearchDirectionCalculator](#) > [SearchDirCalc](#) ()

Static Public Member Functions

- static void [print_copyright_message](#) (const [Journalist](#) &jnlst)
- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions**Default Compiler Generated Methods**

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IpoptAlgorithm](#) ()
Default Constructor.
- [IpoptAlgorithm](#) (const [IpoptAlgorithm](#) &)
Copy Constructor.
- void [operator=](#) (const [IpoptAlgorithm](#) &)
Overloaded Equals Operator.

Main steps of the algorithm

- void [UpdateHessian](#) ()
Method for updating the current Hessian.
- bool [UpdateBarrierParameter](#) ()
Method to update the barrier parameter.
- bool [ComputeSearchDirection](#) ()
Method to setup the call to the [PDSolver](#).
- void [ComputeAcceptableTrialPoint](#) ()
Method computing the new iterate (usually via line search).
- void [AcceptTrialPoint](#) ()
Method for accepting the trial point as the new iteration, possibly after adjusting the variable bounds in the [NLP](#).
- void [OutputIteration](#) ()
Do all the output for one iteration.
- void [InitializeIterates](#) ()

- *Sets up initial values for the iterates, Corrects the initial values for x and s (force in bounds)*
- void `PrintProblemStatistics` ()
Print the problem size statistics.
- void `ComputeFeasibilityMultipliers` ()
Compute the Lagrangian multipliers for a feasibility problem.

auxiliary functions

- void `calc_number_of_bounds` (const `Vector` & x , const `Vector` & x_L , const `Vector` & x_U , const `Matrix` & Px_L , const `Matrix` & Px_U , `Index` & n_{tot} , `Index` & n_{only_lower} , `Index` & n_{both} , `Index` & n_{only_upper})
- `Number correct_bound_multiplier` (const `Vector` & $trial_z$, const `Vector` & $trial_slack$, const `Vector` & $trial_compl$, `SmartPtr`< const `Vector` > & new_trial_z)
Method for ensuring that the trial multipliers are not too far from the primal estimate.

Private Attributes

Strategy objects

- `SmartPtr`
 < `SearchDirectionCalculator` > `search_dir_calculator_`
- `SmartPtr`< `LineSearch` > `line_search_`
- `SmartPtr`< `MuUpdate` > `mu_update_`
- `SmartPtr`< `ConvergenceCheck` > `conv_check_`
- `SmartPtr`< `IterateInitializer` > `iterate_initializer_`
- `SmartPtr`< `IterationOutput` > `iter_output_`
- `SmartPtr`< `HessianUpdater` > `hessian_updater_`
- `SmartPtr`< `EqMultiplierCalculator` > `eq_multiplier_calculator_`
The multiplier calculator (for y_c and y_d) has to be set only if option `recalc_y` is set to true.

internal flags

- bool `skip_print_problem_stats_`
Flag indicating if the statistic should not be printed.

Algorithmic parameters

- `Number kappa_sigma_`
safeguard factor for bound multipliers.
- bool `recalc_y_`
Flag indicating whether the y multipliers should be recalculated with the `eq_multiplier_calculator` object for each new point.
- `Number recalc_y_feas_tol_`
Feasibility threshold for `recalc_y`.
- bool `mehrotra_algorithm_`
Flag indicating if we want to do Mehrotra's algorithm.
- std::string `linear_solver_`
String specifying linear solver.

Additional Inherited Members

6.68.1 Detailed Description

The main ipopt algorithm class.

Main [Ipopt](#) algorithm class, contains the main optimize method, handles the execution of the optimization. The constructor initializes the data structures through the nlp, and the Optimize method then assumes that everything is initialized and ready to go. After an optimization is complete, the user can access the solution through the passed in ip_data structure. Multiple calls to the Optimize method are allowed as long as the structure of the problem remains the same (i.e. starting point or nlp parameter changes only).

Definition at line 45 of file IpoptAlg.hpp.

6.68.2 Constructor & Destructor Documentation

```
6.68.2.1 Ipopt::IpoptAlgorithm::IpoptAlgorithm ( const SmartPtr< SearchDirectionCalculator > & search_dir_calculator,
const SmartPtr< LineSearch > & line_search, const SmartPtr< MuUpdate > & mu_update, const SmartPtr<
ConvergenceCheck > & conv_check, const SmartPtr< IterateInitializer > & iterate_initializer, const SmartPtr<
IterationOutput > & iter_output, const SmartPtr< HessianUpdater > & hessian_updater, const SmartPtr<
EqMultiplierCalculator > & eq_multiplier_calculator = NULL )
```

Constructor.

(The [IpoptAlgorithm](#) uses smart pointers for these passed-in pieces to make sure that a user of IpoptAlgorithm cannot pass in an object created on the stack!)

```
6.68.2.2 virtual Ipopt::IpoptAlgorithm::~IpoptAlgorithm ( ) [virtual]
```

Default destructor.

```
6.68.2.3 Ipopt::IpoptAlgorithm::IpoptAlgorithm ( ) [private]
```

Default Constructor.

```
6.68.2.4 Ipopt::IpoptAlgorithm::IpoptAlgorithm ( const IpoptAlgorithm & ) [private]
```

Copy Constructor.

6.68.3 Member Function Documentation

```
6.68.3.1 virtual bool Ipopt::IpoptAlgorithm::InitializeImpl ( const OptionsList & options, const std::string & prefix )
[virtual]
```

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

```
6.68.3.2 SolverReturn Ipopt::IpoptAlgorithm::Optimize ( bool isResto = false )
```

Main solve method.

```
6.68.3.3 static void Ipopt::IpoptAlgorithm::RegisterOptions ( SmartPtr< RegisteredOptions > roptions ) [static]
```

Methods for IpoptType.

6.68.3.4 **SmartPtr<SearchDirectionCalculator> Ipopt::IpoptAlgorithm::SearchDirCalc ()** [inline]

Definition at line 83 of file IpoptAlg.hpp.

6.68.3.5 **static void Ipopt::IpoptAlgorithm::print_copyright_message (const Journalist & jnlst)** [static]

6.68.3.6 **void Ipopt::IpoptAlgorithm::operator= (const IpoptAlgorithm &)** [private]

Overloaded Equals Operator.

6.68.3.7 **void Ipopt::IpoptAlgorithm::UpdateHessian ()** [private]

Method for updating the current Hessian.

This can either just evaluate the exact Hessian (based on the current iterate), or perform a quasi-Newton update.

6.68.3.8 **bool Ipopt::IpoptAlgorithm::UpdateBarrierParameter ()** [private]

Method to update the barrier parameter.

Returns false, if the algorithm can't continue with the regular procedure and needs to revert to a fallback mechanism in the line search (such as restoration phase)

6.68.3.9 **bool Ipopt::IpoptAlgorithm::ComputeSearchDirection ()** [private]

Method to setup the call to the [PDSysolver](#).

Returns false, if the algorithm can't continue with the regular procedure and needs to revert to a fallback mechanism in the line search (such as restoration phase)

6.68.3.10 **void Ipopt::IpoptAlgorithm::ComputeAcceptableTrialPoint ()** [private]

Method computing the new iterate (usually via line search).

The acceptable point is the one in trial after return.

6.68.3.11 **void Ipopt::IpoptAlgorithm::AcceptTrialPoint ()** [private]

Method for accepting the trial point as the new iteration, possibly after adjusting the variable bounds in the [NLP](#).

6.68.3.12 **void Ipopt::IpoptAlgorithm::OutputIteration ()** [private]

Do all the output for one iteration.

6.68.3.13 **void Ipopt::IpoptAlgorithm::InitializeIterates ()** [private]

Sets up initial values for the iterates, Corrects the initial values for x and s (force in bounds)

6.68.3.14 **void Ipopt::IpoptAlgorithm::PrintProblemStatistics ()** [private]

Print the problem size statistics.

6.68.3.15 **void Ipopt::IpoptAlgorithm::ComputeFeasibilityMultipliers ()** [private]

Compute the Lagrangian multipliers for a feasibility problem.

6.68.3.16 **void Ipopt::IpoptAlgorithm::calc_number_of_bounds (const Vector & x, const Vector & x_L, const Vector & x_U, const Matrix & Px_L, const Matrix & Px_U, Index & n_tot, Index & n_only_lower, Index & n_both, Index & n_only_upper)** [private]

6.68.3.17 **Number** Ipopt::IpoptAlgorithm::correct_bound_multiplier (const Vector & trial_z, const Vector & trial_slack, const Vector & trial_compl, SmartPtr< const Vector > & new_trial_z) [private]

Method for ensuring that the trial multipliers are not too far from the primal estimate.

If a correction is made, new_trial_z is a pointer to the corrected multiplier, and the return value of this method give the magnitude of the largest correction that we done. If no correction was made, new_trial_z is just a pointer to trial_z, and the return value is zero.

6.68.4 Member Data Documentation

6.68.4.1 **SmartPtr<SearchDirectionCalculator>** Ipopt::IpoptAlgorithm::search_dir_calculator_ [private]

Definition at line 112 of file IpoptAlg.hpp.

6.68.4.2 **SmartPtr<LineSearch>** Ipopt::IpoptAlgorithm::line_search_ [private]

Definition at line 113 of file IpoptAlg.hpp.

6.68.4.3 **SmartPtr<MuUpdate>** Ipopt::IpoptAlgorithm::mu_update_ [private]

Definition at line 114 of file IpoptAlg.hpp.

6.68.4.4 **SmartPtr<ConvergenceCheck>** Ipopt::IpoptAlgorithm::conv_check_ [private]

Definition at line 115 of file IpoptAlg.hpp.

6.68.4.5 **SmartPtr<IterateInitializer>** Ipopt::IpoptAlgorithm::iterate_initializer_ [private]

Definition at line 116 of file IpoptAlg.hpp.

6.68.4.6 **SmartPtr<IterationOutput>** Ipopt::IpoptAlgorithm::iter_output_ [private]

Definition at line 117 of file IpoptAlg.hpp.

6.68.4.7 **SmartPtr<HessianUpdater>** Ipopt::IpoptAlgorithm::hessian_updater_ [private]

Definition at line 118 of file IpoptAlg.hpp.

6.68.4.8 **SmartPtr<EqMultiplierCalculator>** Ipopt::IpoptAlgorithm::eq_multiplier_calculator_ [private]

The multiplier calculator (for y_c and y_d) has to be set only if option recalc_y is set to true.

Definition at line 121 of file IpoptAlg.hpp.

6.68.4.9 **bool** Ipopt::IpoptAlgorithm::skip_print_problem_stats_ [private]

Flag indicating if the statistic should not be printed.

Definition at line 171 of file IpoptAlg.hpp.

6.68.4.10 **Number** Ipopt::IpoptAlgorithm::kappa_sigma_ [private]

safeguard factor for bound multipliers.

If value ≥ 1 , then the dual variables will never deviate from the primal estimate by more than the factors kappa_sigma and $1./\text{kappa_sigma}$.

Definition at line 180 of file IpoptAlg.hpp.

6.68.4.11 `bool Ipopt::IpoptAlgorithm::recalc_y_ [private]`

Flag indicating whether the y multipliers should be recalculated with the `eq_multiplier_calculator` object for each new point.

Definition at line 184 of file `IpIpoptAlg.hpp`.

6.68.4.12 `Number Ipopt::IpoptAlgorithm::recalc_y_feas_tol_ [private]`

Feasibility threshold for `recalc_y`.

Definition at line 186 of file `IpIpoptAlg.hpp`.

6.68.4.13 `bool Ipopt::IpoptAlgorithm::mehrotra_algorithm_ [private]`

Flag indicating if we want to do Mehrotra's algorithm.

This means that a number of options are ignored, or have to be set (or are automatically set) to certain values.

Definition at line 190 of file `IpIpoptAlg.hpp`.

6.68.4.14 `std::string Ipopt::IpoptAlgorithm::linear_solver_ [private]`

String specifying linear solver.

Definition at line 192 of file `IpIpoptAlg.hpp`.

The documentation for this class was generated from the following file:

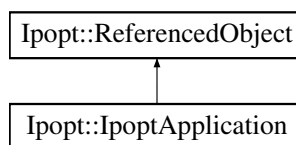
- [Algorithm/IpIpoptAlg.hpp](#)

6.69 Ipopt::IpoptApplication Class Reference

This is the main application class for making calls to [Ipopt](#).

```
#include <IpIpoptApplication.hpp>
```

Inheritance diagram for `Ipopt::IpoptApplication`:



Public Member Functions

- [IpoptApplication](#) (`bool create_console_out=true`, `bool create_empty=false`)
- [IpoptApplication](#) ([SmartPtr](#)< [RegisteredOptions](#) > `reg_options`, [SmartPtr](#)< [OptionsList](#) > `options`, [SmartPtr](#)< [Journalist](#) > `jnlst`)

Another constructor that assumes that the code in the (default) constructor has already been executed.

- `virtual ~IpoptApplication ()`
- `virtual SmartPtr`
`< IpoptApplication > clone ()`

Method for creating a new IpoptApplication that uses the same journalist and registered options, and a copy of the options list.

- virtual [ApplicationReturnStatus Initialize](#) (std::istream &is)
Initialization method.
- virtual [ApplicationReturnStatus Initialize](#) (std::string params_file)
Initialization method.
- virtual [ApplicationReturnStatus Initialize](#) ()
Initialize method.
- virtual bool [OpenOutputFile](#) (std::string file_name, EJournalLevel print_level)
Method for opening an output file with given print_level.
- void [PrintCopyrightMessage](#) ()
Method for printing [Ipopt](#) copyright message now instead of just before the optimization.
- void [RethrowNonIpoptException](#) (bool dorethrow)
Method to set whether non-ipopt non-bad_alloc exceptions are rethrown by [Ipopt](#).

Solve methods

- virtual [ApplicationReturnStatus OptimizeTNLP](#) (const [SmartPtr](#)< [TNLP](#) > &tnlp)
Solve a problem that inherits from [TNLP](#).
- virtual [ApplicationReturnStatus OptimizeNLP](#) (const [SmartPtr](#)< [NLP](#) > &nlp)
Solve a problem that inherits from [NLP](#).
- virtual [ApplicationReturnStatus OptimizeNLP](#) (const [SmartPtr](#)< [NLP](#) > &nlp, [SmartPtr](#)< [AlgorithmBuilder](#) > &alg_builder)
Solve a problem that inherits from [NLP](#).
- virtual [ApplicationReturnStatus ReOptimizeTNLP](#) (const [SmartPtr](#)< [TNLP](#) > &tnlp)
Solve a problem (that inherits from [TNLP](#)) for a repeated time.
- virtual [ApplicationReturnStatus ReOptimizeNLP](#) (const [SmartPtr](#)< [NLP](#) > &nlp)
Solve a problem (that inherits from [NLP](#)) for a repeated time.

Accessor methods

- virtual [SmartPtr](#)< [Journalist](#) > [Jnlst](#) ()
Get the [Journalist](#) for printing output.
- virtual [SmartPtr](#)
< [RegisteredOptions](#) > [RegOptions](#) ()
Get a pointer to [RegisteredOptions](#) object to add new options.
- virtual [SmartPtr](#)< [OptionsList](#) > [Options](#) ()
Get the options list for setting options.
- virtual [SmartPtr](#)< const
[OptionsList](#) > [Options](#) () const
Get the options list for setting options (const version)
- virtual [SmartPtr](#)< [SolveStatistics](#) > [Statistics](#) ()
Get the object with the statistics about the most recent optimization run.
- virtual [SmartPtr](#)< [IpoptNLP](#) > [IpoptNLObject](#) ()
Get the [IpoptNLP](#) Object.
- [SmartPtr](#)< [IpoptData](#) > [IpoptDataObject](#) ()
Get the [IpoptData](#) Object.
- virtual [SmartPtr](#)
< [IpoptCalculatedQuantities](#) > [IpoptCQObject](#) ()
Get the [IpoptCQ](#) Object.
- [SmartPtr](#)< [IpoptAlgorithm](#) > [AlgorithmObject](#) ()
Get the Algorithm Object.

Static Public Member Functions

- static void [RegisterAllIpoptOptions](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)
Method to registering all [Ipopt](#) options.

Methods for IpoptTypeInfo

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)

Private Member Functions

- [ApplicationReturnStatus](#) [call_optimize](#) ()
Method for the actual optimize call of the [Ipopt](#) algorithm.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IpoptApplication](#) (const [IpoptApplication](#) &)
Default Constructor.
- void [operator=](#) (const [IpoptApplication](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [Journalist](#) > [jnlst_](#)
[Journalist](#) for reporting output.
- [SmartPtr](#)< [RegisteredOptions](#) > [reg_options_](#)
[RegisteredOptions](#).
- [SmartPtr](#)< [OptionsList](#) > [options_](#)
[OptionsList](#) used for the application.
- [SmartPtr](#)< [SolveStatistics](#) > [statistics_](#)
Object for storing statistics about the most recent optimization run.
- [SmartPtr](#)< [IpoptAlgorithm](#) > [alg_](#)
Object with the algorithm skeleton.
- [SmartPtr](#)< [IpoptNLP](#) > [ip_nlp_](#)
[IpoptNLP](#) Object for the [NLP](#).
- [SmartPtr](#)< [IpoptData](#) > [ip_data_](#)
[IpoptData](#) Object for the [NLP](#).
- [SmartPtr](#)
< [IpoptCalculatedQuantities](#) > [ip_cq_](#)
[IpoptCalculatedQuantities](#) Object for the [NLP](#).
- [SmartPtr](#)< [NLP](#) > [nlp_adapter_](#)
Pointer to the [TNLPAdapter](#) used to convert the [TNLP](#) to an [NLP](#).

Variables that customize the application behavior

- bool [read_params_dat_](#)

- *Decide whether or not the `ipopt.opt` file should be read.*
- bool [rethrow_nonipoptexception_](#)
Decide whether non-ipopt non-bad_alloc exceptions should be rethrown.

Algorithmic parameters

- bool [inexact_algorithm_](#)
Flag indicating if we are to use the inexact linear solver option.
- bool [replace_bounds_](#)
Flag indicating if all bounds should be replaced by inequality constraints.

6.69.1 Detailed Description

This is the main application class for making calls to [Ipopt](#).

Definition at line 47 of file `IpoptApplication.hpp`.

6.69.2 Constructor & Destructor Documentation

6.69.2.1 `Ipopt::IpoptApplication::IpoptApplication (bool create_console_out = true, bool create_empty = false)`

6.69.2.2 `Ipopt::IpoptApplication::IpoptApplication (SmartPtr< RegisteredOptions > reg_options, SmartPtr< OptionsList > options, SmartPtr< Journalist > jnlst)`

Another constructor that assumes that the code in the (default) constructor has already been executed.

6.69.2.3 `virtual Ipopt::IpoptApplication::~~IpoptApplication () [virtual]`

6.69.2.4 `Ipopt::IpoptApplication::IpoptApplication (const IpoptApplication &) [private]`

Default Constructor.

Copy Constructor

6.69.3 Member Function Documentation

6.69.3.1 `virtual SmartPtr<IpoptApplication> Ipopt::IpoptApplication::clone () [virtual]`

Method for creating a new `IpoptApplication` that uses the same `journalist` and registered options, and a copy of the options

list.

6.69.3.2 `virtual ApplicationReturnStatus Ipopt::IpoptApplication::Initialize (std::istream & is) [virtual]`

Initialization method.

This method reads options from the input stream and initializes the journalists. It returns something other than `Solve_ Succeeded` if there was a problem in the initialization (such as an invalid option). You should call one of the initialization methods at some point before the first optimize call.

6.69.3.3 `virtual ApplicationReturnStatus Ipopt::IpoptApplication::Initialize (std::string params_file) [virtual]`

Initialization method.

This method reads options from the params file and initializes the journalists. It returns something other than `Solve_Succeeded` if there was a problem in the initialization (such as an invalid option). You should call one of the initialization methods at some point before the first optimize call. Note: You can skip the processing of a params file by setting `params_file` to "".

6.69.3.4 `virtual ApplicationReturnStatus lpopt::lpoptApplication::Initialize () [virtual]`

Initialize method.

This method reads the options file specified by the `option_file_name` option and initializes the journalists. You should call this method at some point before the first optimize call. It returns something other than `Solve_Succeeded` if there was a problem in the initialization (such as an invalid option).

6.69.3.5 `virtual ApplicationReturnStatus lpopt::lpoptApplication::OptimizeTNLP (const SmartPtr< TNLP > & tnlp) [virtual]`

Solve a problem that inherits from [TNLP](#).

6.69.3.6 `virtual ApplicationReturnStatus lpopt::lpoptApplication::OptimizeNLP (const SmartPtr< NLP > & nlp) [virtual]`

Solve a problem that inherits from [NLP](#).

6.69.3.7 `virtual ApplicationReturnStatus lpopt::lpoptApplication::OptimizeNLP (const SmartPtr< NLP > & nlp, SmartPtr< AlgorithmBuilder > & alg_builder) [virtual]`

Solve a problem that inherits from [NLP](#).

6.69.3.8 `virtual ApplicationReturnStatus lpopt::lpoptApplication::ReOptimizeTNLP (const SmartPtr< TNLP > & tnlp) [virtual]`

Solve a problem (that inherits from [TNLP](#)) for a repeated time.

The `OptimizeTNLP` method must have been called before. The [TNLP](#) must be the same object, and the structure (number of variables and constraints and position of nonzeros in Jacobian and Hessian must be the same).

6.69.3.9 `virtual ApplicationReturnStatus lpopt::lpoptApplication::ReOptimizeNLP (const SmartPtr< NLP > & nlp) [virtual]`

Solve a problem (that inherits from [NLP](#)) for a repeated time.

The `OptimizeNLP` method must have been called before. The [NLP](#) must be the same object, and the structure (number of variables and constraints and position of nonzeros in Jacobian and Hessian must be the same).

6.69.3.10 `virtual bool lpopt::lpoptApplication::OpenOutputFile (std::string file_name, EJournalLevel print_level) [virtual]`

Method for opening an output file with given `print_level`.

Returns false if there was a problem.

6.69.3.11 `virtual SmartPtr< Journalist > lpopt::lpoptApplication::Jnlst () [inline], [virtual]`

Get the [Journalist](#) for printing output.

Definition at line 126 of file `lpoptApplication.hpp`.

6.69.3.12 `virtual SmartPtr<RegisteredOptions> Ipopt::IpoptApplication::RegOptions () [inline],[virtual]`

Get a pointer to [RegisteredOptions](#) object to add new options.

Definition at line 133 of file `IpoptApplication.hpp`.

6.69.3.13 `virtual SmartPtr<OptionsList> Ipopt::IpoptApplication::Options () [inline],[virtual]`

Get the options list for setting options.

Definition at line 139 of file `IpoptApplication.hpp`.

6.69.3.14 `virtual SmartPtr<const OptionsList> Ipopt::IpoptApplication::Options () const [inline],[virtual]`

Get the options list for setting options (const version)

Definition at line 145 of file `IpoptApplication.hpp`.

6.69.3.15 `virtual SmartPtr<SolveStatistics> Ipopt::IpoptApplication::Statistics () [virtual]`

Get the object with the statistics about the most recent optimization run.

6.69.3.16 `virtual SmartPtr<IpoptNLP> Ipopt::IpoptApplication::IpoptNLPObject () [virtual]`

Get the [IpoptNLP](#) Object.

6.69.3.17 `SmartPtr<IpoptData> Ipopt::IpoptApplication::IpoptDataObject ()`

Get the [IpoptData](#) Object.

6.69.3.18 `virtual SmartPtr<IpoptCalculatedQuantities> Ipopt::IpoptApplication::IpoptCQObject () [virtual]`

Get the [IpoptCQ](#) Object.

6.69.3.19 `SmartPtr<IpoptAlgorithm> Ipopt::IpoptApplication::AlgorithmObject ()`

Get the Algorithm Object.

6.69.3.20 `void Ipopt::IpoptApplication::PrintCopyrightMessage ()`

Method for printing [Ipopt](#) copyright message now instead of just before the optimization.

If you want to have the copy right message printed earlier than by default, call this method at the convenient time.

6.69.3.21 `void Ipopt::IpoptApplication::RethrowNonIpoptException (bool dorethrow) [inline]`

Method to set whether non-ipopt non-`bad_alloc` exceptions are rethrown by [Ipopt](#).

By default, non-`Ipopt` and non-`std::bad_alloc` exceptions are caught by `Ipopt`s initialization and optimization methods and the status `NonIpopt_Exception_Thrown` is returned. This function allows to enable rethrowing of such exceptions.

Definition at line 180 of file `IpoptApplication.hpp`.

6.69.3.22 `static void Ipopt::IpoptApplication::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

6.69.3.23 `static void Ipopt::IpoptApplication::RegisterAllIpoptOptions (const SmartPtr< RegisteredOptions > & roptions) [static]`

Method to registering all [Ipopt](#) options.

6.69.3.24 `void Ipopt::IpoptApplication::operator= (const IpoptApplication &) [private]`

Overloaded Equals Operator.

6.69.3.25 `ApplicationReturnStatus Ipopt::IpoptApplication::call_optimize () [private]`

Method for the actual optimize call of the [Ipopt](#) algorithm.

This is used both for Optimize and ReOptimize

6.69.4 Member Data Documentation

6.69.4.1 `bool Ipopt::IpoptApplication::read_params_dat_ [private]`

Decide whether or not the ipopt.opt file should be read.

Definition at line 220 of file `IpoptApplication.hpp`.

6.69.4.2 `bool Ipopt::IpoptApplication::rethrow_nonipoptexception_ [private]`

Decide whether non-ipopt non-bad_alloc exceptions should be rethrown.

Definition at line 223 of file `IpoptApplication.hpp`.

6.69.4.3 `SmartPtr<Journalist> Ipopt::IpoptApplication::jnlst_ [private]`

[Journalist](#) for reporting output.

Definition at line 227 of file `IpoptApplication.hpp`.

6.69.4.4 `SmartPtr<RegisteredOptions> Ipopt::IpoptApplication::reg_options_ [private]`

[RegisteredOptions](#).

Definition at line 230 of file `IpoptApplication.hpp`.

6.69.4.5 `SmartPtr<OptionsList> Ipopt::IpoptApplication::options_ [private]`

[OptionsList](#) used for the application.

Definition at line 233 of file `IpoptApplication.hpp`.

6.69.4.6 `SmartPtr<SolveStatistics> Ipopt::IpoptApplication::statistics_ [private]`

Object for storing statistics about the most recent optimization run.

Definition at line 237 of file `IpoptApplication.hpp`.

6.69.4.7 `SmartPtr<IpoptAlgorithm> Ipopt::IpoptApplication::alg_ [private]`

Object with the algorithm skeleton.

Definition at line 241 of file `IpoptApplication.hpp`.

6.69.4.8 `SmartPtr<IpoptNLP> Ipopt::IpoptApplication::ip_nlp_ [private]`

[IpoptNLP](#) Object for the [NLP](#).

We keep this around for a ReOptimize warm start.

Definition at line 245 of file `IpoptApplication.hpp`.

6.69.4.9 **SmartPointer<IpoptData>** Ipopt::IpoptApplication::ip_data_ [private]

[IpoptData](#) Object for the [NLP](#).

We keep this around for a ReOptimize warm start.

Definition at line 250 of file IpoptApplication.hpp.

6.69.4.10 **SmartPointer<IpoptCalculatedQuantities>** Ipopt::IpoptApplication::ip_cq_ [private]

[IpoptCalculatedQuantities](#) Object for the [NLP](#).

We keep this around for a ReOptimize warm start.

Definition at line 255 of file IpoptApplication.hpp.

6.69.4.11 **SmartPointer<NLP>** Ipopt::IpoptApplication::nlp_adapter_ [private]

Pointer to the [TNLPAdapter](#) used to convert the [TNLP](#) to an [NLP](#).

We keep this around for the ReOptimizerTNLP call.

Definition at line 259 of file IpoptApplication.hpp.

6.69.4.12 **bool** Ipopt::IpoptApplication::inexact_algorithm_ [private]

Flag indicating if we are to use the inexact linear solver option.

Definition at line 264 of file IpoptApplication.hpp.

6.69.4.13 **bool** Ipopt::IpoptApplication::replace_bounds_ [private]

Flag indicating if all bounds should be replaced by inequality constraints.

This is necessary for the inexact algorithm.

Definition at line 267 of file IpoptApplication.hpp.

The documentation for this class was generated from the following file:

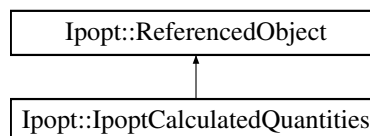
- [Interfaces/IpoptApplication.hpp](#)

6.70 Ipopt::IpoptCalculatedQuantities Class Reference

Class for all IPOPT specific calculated quantities.

```
#include <IpIpoptCalculatedQuantities.hpp>
```

Inheritance diagram for Ipopt::IpoptCalculatedQuantities:



Public Member Functions

- void [SetAddCq](#) ([SmartPointer< IpoptAdditionalCq >](#) add_cq)

Method for setting pointer for additional calculated quantities.

- `bool HaveAddCq ()`
Method detecting if additional object for calculated quantities has already been set.
- `bool Initialize (const Journalist &jnlst, const OptionsList &options, const std::string &prefix)`
This method must be called to initialize the global algorithmic parameters.
- `Number curr_avg_compl ()`
average of current values of the complementarities
- `Number trial_avg_compl ()`
average of trial values of the complementarities
- `Number curr_gradBarrTDelta ()`
inner_product of current barrier obj.
- `Number CalcNormOfType (ENormType NormType, std::vector< SmartPtr< const Vector > > vecs)`
Compute the norm of a specific type of a set of vectors (uncached)
- `Number CalcNormOfType (ENormType NormType, const Vector &vec1, const Vector &vec2)`
Compute the norm of a specific type of two vectors (uncached)
- `ENormType constr_viol_normtype () const`
Norm type used for calculating constraint violation.
- `bool IsSquareProblem () const`
Method returning true if this is a square problem.
- `SmartPtr< lpoptNLP > & GetlpoptNLP ()`
Method returning the lpoptNLP object.
- `lpoptAdditionalCq & AdditionalCq ()`

Constructors/Destructors

- `lpoptCalculatedQuantities (const SmartPtr< lpoptNLP > &ip_nlp, const SmartPtr< lpoptData > &ip_data)`
Constructor.
- `virtual ~lpoptCalculatedQuantities ()`
Default destructor.

Slacks

- `SmartPtr< const Vector > curr_slack_x_L ()`
Slacks for x_L (at current iterate)
- `SmartPtr< const Vector > curr_slack_x_U ()`
Slacks for x_U (at current iterate)
- `SmartPtr< const Vector > curr_slack_s_L ()`
Slacks for s_L (at current iterate)
- `SmartPtr< const Vector > curr_slack_s_U ()`
Slacks for s_U (at current iterate)
- `SmartPtr< const Vector > trial_slack_x_L ()`
Slacks for x_L (at trial point)
- `SmartPtr< const Vector > trial_slack_x_U ()`
Slacks for x_U (at trial point)
- `SmartPtr< const Vector > trial_slack_s_L ()`
Slacks for s_L (at trial point)
- `SmartPtr< const Vector > trial_slack_s_U ()`
Slacks for s_U (at trial point)
- `Index AdjustedTrialSlacks ()`
Indicating whether or not we "fudged" the slacks.
- `void ResetAdjustedTrialSlacks ()`

Reset the flags for "fudged" slacks.

Objective function

- virtual [Number curr_f \(\)](#)
Value of objective function (at current point)
- virtual [Number unscaled_curr_f \(\)](#)
Unscaled value of the objective function (at the current point)
- virtual [Number trial_f \(\)](#)
Value of objective function (at trial point)
- virtual [Number unscaled_trial_f \(\)](#)
Unscaled value of the objective function (at the trial point)
- [SmartPtr< const Vector > curr_grad_f \(\)](#)
Gradient of objective function (at current point)
- [SmartPtr< const Vector > trial_grad_f \(\)](#)
Gradient of objective function (at trial point)

Barrier Objective Function

- virtual [Number curr_barrier_obj \(\)](#)
Barrier Objective Function Value (at current iterate with current mu)
- virtual [Number trial_barrier_obj \(\)](#)
Barrier Objective Function Value (at trial point with current mu)
- [SmartPtr< const Vector > curr_grad_barrier_obj_x \(\)](#)
Gradient of barrier objective function with respect to x (at current point with current mu)
- [SmartPtr< const Vector > curr_grad_barrier_obj_s \(\)](#)
Gradient of barrier objective function with respect to s (at current point with current mu)
- [SmartPtr< const Vector > grad_kappa_times_damping_x \(\)](#)
Gradient of the damping term with respect to x (times kappa_d)
- [SmartPtr< const Vector > grad_kappa_times_damping_s \(\)](#)
Gradient of the damping term with respect to s (times kappa_d)

Constraints

- [SmartPtr< const Vector > curr_c \(\)](#)
c(x) (at current point)
- [SmartPtr< const Vector > unscaled_curr_c \(\)](#)
unscaled c(x) (at current point)
- [SmartPtr< const Vector > trial_c \(\)](#)
c(x) (at trial point)
- [SmartPtr< const Vector > unscaled_trial_c \(\)](#)
unscaled c(x) (at trial point)
- [SmartPtr< const Vector > curr_d \(\)](#)
d(x) (at current point)
- [SmartPtr< const Vector > unscaled_curr_d \(\)](#)
unscaled d(x) (at current point)
- [SmartPtr< const Vector > trial_d \(\)](#)
d(x) (at trial point)
- [SmartPtr< const Vector > curr_d_minus_s \(\)](#)
d(x) - s (at current point)
- [SmartPtr< const Vector > trial_d_minus_s \(\)](#)
d(x) - s (at trial point)
- [SmartPtr< const Matrix > curr_jac_c \(\)](#)
Jacobian of c (at current point)

- `SmartPtr< const Matrix > trial_jac_c ()`
Jacobian of c (at trial point)
- `SmartPtr< const Matrix > curr_jac_d ()`
Jacobian of d (at current point)
- `SmartPtr< const Matrix > trial_jac_d ()`
Jacobian of d (at trial point)
- `SmartPtr< const Vector > curr_jac_cT_times_vec (const Vector &vec)`
Product of Jacobian (evaluated at current point) of C transpose with general vector.
- `SmartPtr< const Vector > trial_jac_cT_times_vec (const Vector &vec)`
Product of Jacobian (evaluated at trial point) of C transpose with general vector.
- `SmartPtr< const Vector > curr_jac_dT_times_vec (const Vector &vec)`
Product of Jacobian (evaluated at current point) of D transpose with general vector.
- `SmartPtr< const Vector > trial_jac_dT_times_vec (const Vector &vec)`
Product of Jacobian (evaluated at trial point) of D transpose with general vector.
- `SmartPtr< const Vector > curr_jac_cT_times_curr_y_c ()`
Product of Jacobian (evaluated at current point) of C transpose with current y_c.
- `SmartPtr< const Vector > trial_jac_cT_times_trial_y_c ()`
Product of Jacobian (evaluated at trial point) of C transpose with trial y_c.
- `SmartPtr< const Vector > curr_jac_dT_times_curr_y_d ()`
Product of Jacobian (evaluated at current point) of D transpose with current y_d.
- `SmartPtr< const Vector > trial_jac_dT_times_trial_y_d ()`
Product of Jacobian (evaluated at trial point) of D transpose with trial y_d.
- `SmartPtr< const Vector > curr_jac_c_times_vec (const Vector &vec)`
Product of Jacobian (evaluated at current point) of C with general vector.
- `SmartPtr< const Vector > curr_jac_d_times_vec (const Vector &vec)`
Product of Jacobian (evaluated at current point) of D with general vector.
- `virtual Number curr_constraint_violation ()`
Constraint Violation (at current iterate).
- `virtual Number trial_constraint_violation ()`
Constraint Violation (at trial point).
- `virtual Number curr_nlp_constraint_violation (ENormType NormType)`
Real constraint violation in a given norm (at current iterate).
- `virtual Number unscaled_curr_nlp_constraint_violation (ENormType NormType)`
Unscaled real constraint violation in a given norm (at current iterate).
- `virtual Number unscaled_trial_nlp_constraint_violation (ENormType NormType)`
Unscaled real constraint violation in a given norm (at trial iterate).

Hessian matrices

- `SmartPtr< const SymMatrix > curr_exact_hessian ()`
exact Hessian at current iterate (uncached)

primal-dual error and its components

- `SmartPtr< const Vector > curr_grad_lag_x ()`
x-part of gradient of Lagrangian function (at current point)
- `SmartPtr< const Vector > trial_grad_lag_x ()`
x-part of gradient of Lagrangian function (at trial point)
- `SmartPtr< const Vector > curr_grad_lag_s ()`
s-part of gradient of Lagrangian function (at current point)
- `SmartPtr< const Vector > trial_grad_lag_s ()`
s-part of gradient of Lagrangian function (at trial point)
- `SmartPtr< const Vector > curr_grad_lag_with_damping_x ()`
*x-part of gradient of Lagrangian function (at current point)
including linear damping term*

- [SmartPtr< const Vector > curr_grad_lag_with_damping_s \(\)](#)
s-part of gradient of Lagrangian function (at current point)
including linear damping term
- [SmartPtr< const Vector > curr_compl_x_L \(\)](#)
Complementarity for x_L (for current iterate)
- [SmartPtr< const Vector > curr_compl_x_U \(\)](#)
Complementarity for x_U (for current iterate)
- [SmartPtr< const Vector > curr_compl_s_L \(\)](#)
Complementarity for s_L (for current iterate)
- [SmartPtr< const Vector > curr_compl_s_U \(\)](#)
Complementarity for s_U (for current iterate)
- [SmartPtr< const Vector > trial_compl_x_L \(\)](#)
Complementarity for x_L (for trial iterate)
- [SmartPtr< const Vector > trial_compl_x_U \(\)](#)
Complementarity for x_U (for trial iterate)
- [SmartPtr< const Vector > trial_compl_s_L \(\)](#)
Complementarity for s_L (for trial iterate)
- [SmartPtr< const Vector > trial_compl_s_U \(\)](#)
Complementarity for s_U (for trial iterate)
- [SmartPtr< const Vector > curr_relaxed_compl_x_L \(\)](#)
Relaxed complementarity for x_L (for current iterate and current mu)
- [SmartPtr< const Vector > curr_relaxed_compl_x_U \(\)](#)
Relaxed complementarity for x_U (for current iterate and current mu)
- [SmartPtr< const Vector > curr_relaxed_compl_s_L \(\)](#)
Relaxed complementarity for s_L (for current iterate and current mu)
- [SmartPtr< const Vector > curr_relaxed_compl_s_U \(\)](#)
Relaxed complementarity for s_U (for current iterate and current mu)
- virtual [Number curr_primal_infeasibility \(ENormType NormType\)](#)
Primal infeasibility in a given norm (at current iterate).
- virtual [Number trial_primal_infeasibility \(ENormType NormType\)](#)
Primal infeasibility in a given norm (at trial point)
- virtual [Number curr_dual_infeasibility \(ENormType NormType\)](#)
Dual infeasibility in a given norm (at current iterate)
- virtual [Number trial_dual_infeasibility \(ENormType NormType\)](#)
Dual infeasibility in a given norm (at trial iterate)
- virtual [Number unscaled_curr_dual_infeasibility \(ENormType NormType\)](#)
Unscaled dual infeasibility in a given norm (at current iterate)
- virtual [Number curr_complementarity \(Number mu, ENormType NormType\)](#)
Complementarity (for all complementarity conditions together) in a given norm (at current iterate)
- virtual [Number trial_complementarity \(Number mu, ENormType NormType\)](#)
Complementarity (for all complementarity conditions together) in a given norm (at trial iterate)
- virtual [Number unscaled_curr_complementarity \(Number mu, ENormType NormType\)](#)
Complementarity (for all complementarity conditions together) in a given norm (at current iterate) without NLP scaling.
- [Number CalcCentralityMeasure](#) (const [Vector](#) &compl_x_L, const [Vector](#) &compl_x_U, const [Vector](#) &compl_s_L, const [Vector](#) &compl_s_U)
Centrality measure (in spirit of the -infinity-neighborhood).
- virtual [Number curr_centrality_measure \(\)](#)
Centrality measure at current point.
- virtual [Number curr_nlp_error \(\)](#)
Total optimality error for the original NLP at the current iterate, using scaling factors based on multipliers.
- virtual [Number unscaled_curr_nlp_error \(\)](#)
Total optimality error for the original NLP at the current iterate, but using no scaling based on multipliers, and no scaling for the NLP.
- virtual [Number curr_barrier_error \(\)](#)
Total optimality error for the barrier problem at the current iterate, using scaling factors based on multipliers.

- virtual [Number curr_primal_dual_system_error](#) ([Number](#) mu)
Norm of the primal-dual system for a given mu (at current iterate).
- virtual [Number trial_primal_dual_system_error](#) ([Number](#) mu)
Norm of the primal-dual system for a given mu (at trial iterate).

Computing fraction-to-the-boundary step sizes

- [Number primal_frac_to_the_bound](#) ([Number](#) tau, const [Vector](#) &delta_x, const [Vector](#) &delta_s)
Fraction to the boundary from (current) primal variables x and s for a given step.
- [Number curr_primal_frac_to_the_bound](#) ([Number](#) tau)
Fraction to the boundary from (current) primal variables x and s for internal (current) step.
- [Number dual_frac_to_the_bound](#) ([Number](#) tau, const [Vector](#) &delta_z_L, const [Vector](#) &delta_z_U, const [Vector](#) &delta_v_L, const [Vector](#) &delta_v_U)
Fraction to the boundary from (current) dual variables z and v for a given step.
- [Number uncached_dual_frac_to_the_bound](#) ([Number](#) tau, const [Vector](#) &delta_z_L, const [Vector](#) &delta_z_U, const [Vector](#) &delta_v_L, const [Vector](#) &delta_v_U)
Fraction to the boundary from (current) dual variables z and v for a given step, without caching.
- [Number curr_dual_frac_to_the_bound](#) ([Number](#) tau)
Fraction to the boundary from (current) dual variables z and v for internal (current) step.
- [Number uncached_slack_frac_to_the_bound](#) ([Number](#) tau, const [Vector](#) &delta_x_L, const [Vector](#) &delta_x_U, const [Vector](#) &delta_s_L, const [Vector](#) &delta_s_U)
Fraction to the boundary from (current) slacks for a given step in the slacks.

Sigma matrices

- [SmartPtr](#)< const [Vector](#) > [curr_sigma_x](#) ()
- [SmartPtr](#)< const [Vector](#) > [curr_sigma_s](#) ()

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IpoptCalculatedQuantities](#) ()
Default Constructor.
- [IpoptCalculatedQuantities](#) (const [IpoptCalculatedQuantities](#) &)
Copy Constructor.
- void [operator=](#) (const [IpoptCalculatedQuantities](#) &)
Overloaded Equals Operator.

Auxiliary functions

- [SmartPtr](#)< [Vector](#) > [CalcSlack_L](#) (const [Matrix](#) &P, const [Vector](#) &x, const [Vector](#) &x_bound)
Compute new vector containing the slack to a lower bound (uncached)
- [SmartPtr](#)< [Vector](#) > [CalcSlack_U](#) (const [Matrix](#) &P, const [Vector](#) &x, const [Vector](#) &x_bound)

- Compute new vector containing the slack to a upper bound (uncached)*
- **Number** CalcBarrierTerm (**Number** mu, const **Vector** &slack_x_L, const **Vector** &slack_x_U, const **Vector** &slack_s_L, const **Vector** &slack_s_U)
- Compute barrier term at given point (uncached)*
- **SmartPtr**< const **Vector** > CalcCompl (const **Vector** &slack, const **Vector** &mult)
- Compute complementarity for slack / multiplier pair.*
- **Number** CalcFracToBound (const **Vector** &slack_L, **Vector** &tmp_L, const **Matrix** &P_L, const **Vector** &slack_U, **Vector** &tmp_U, const **Matrix** &P_U, const **Vector** &delta, **Number** tau)
- Compute fraction to the boundary parameter for lower and upper bounds.*
- void ComputeOptimalityErrorScaling (const **Vector** &y_c, const **Vector** &y_d, const **Vector** &z_L, const **Vector** &z_U, const **Vector** &v_L, const **Vector** &v_U, **Number** s_max, **Number** &s_d, **Number** &s_c)
- Compute the scaling factors for the optimality error.*
- **Index** CalculateSafeSlack (**SmartPtr**< **Vector** > &slack, const **SmartPtr**< const **Vector** > &bound, const **SmartPtr**< const **Vector** > &curr_point, const **SmartPtr**< const **Vector** > &multiplier)
- Check if slacks are becoming too small.*
- void ComputeDampingIndicators (**SmartPtr**< const **Vector** > &dampind_x_L, **SmartPtr**< const **Vector** > &dampind_x_U, **SmartPtr**< const **Vector** > &dampind_s_L, **SmartPtr**< const **Vector** > &dampind_s_U)
- Computes the indicator vectors that can be used to filter out those entries in the slack...*
- bool in_restoration_phase ()
- Check if we are in the restoration phase.*

Private Attributes

- **CachedResults**< **SmartPtr**< const **SymMatrix** > > curr_exact_hessian_cache_
- Cache for the exact Hessian.*
- **CachedResults**< **Number** > curr_avg_compl_cache_
- Cache for average of current complementarity.*
- **CachedResults**< **Number** > trial_avg_compl_cache_
- Cache for average of trial complementarity.*
- **CachedResults**< **Number** > curr_gradBarrTDelta_cache_
- Cache for grad barrier obj.*
- bool initialize_called_
- flag indicating if Initialize method has been called (for debugging)*

Pointers for easy access to data and NLP information

- **SmartPtr**< **IpoptNLP** > ip_nlp_
- Ipopt NLP object.*
- **SmartPtr**< **IpoptData** > ip_data_
- Ipopt Data object.*
- **SmartPtr**< **IpoptAdditionalCq** > add_cq_
- Chen-Goldfarb specific calculated quantities.*

Algorithmic Parameters that can be set through the

options list.

Those parameters are initialize by calling the Initialize method.

- **Number** s_max_
- Parameter in formula for computing overall primal-dual optimality error.*
- **Number** kappa_d_
- Weighting factor for the linear damping term added to the barrier objective function.*

- [Number slack_move_](#)
fractional movement allowed in bounds
- [ENormType constr_viol_normtype_](#)
Norm type to be used when calculating the constraint violation.
- [bool warm_start_same_structure_](#)
*Flag indicating whether the *TNLP* with identical structure has already been solved before.*
- [Number mu_target_](#)
Desired value of the barrier parameter.

Caches for slacks

- [CachedResults< SmartPtr< Vector > > curr_slack_x_L_cache_](#)
- [CachedResults< SmartPtr< Vector > > curr_slack_x_U_cache_](#)
- [CachedResults< SmartPtr< Vector > > curr_slack_s_L_cache_](#)
- [CachedResults< SmartPtr< Vector > > curr_slack_s_U_cache_](#)
- [CachedResults< SmartPtr< Vector > > trial_slack_x_L_cache_](#)
- [CachedResults< SmartPtr< Vector > > trial_slack_x_U_cache_](#)
- [CachedResults< SmartPtr< Vector > > trial_slack_s_L_cache_](#)
- [CachedResults< SmartPtr< Vector > > trial_slack_s_U_cache_](#)
- [Index num_adjusted_slack_x_L_](#)
- [Index num_adjusted_slack_x_U_](#)
- [Index num_adjusted_slack_s_L_](#)
- [Index num_adjusted_slack_s_U_](#)

Cached for objective function stuff

- [CachedResults< Number > curr_f_cache_](#)
- [CachedResults< Number > trial_f_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_grad_f_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_grad_f_cache_](#)

Caches for barrier function stuff

- [CachedResults< Number > curr_barrier_obj_cache_](#)
- [CachedResults< Number > trial_barrier_obj_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_grad_barrier_obj_x_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_grad_barrier_obj_s_cache_](#)
- [CachedResults< SmartPtr< const Vector > > grad_kappa_times_damping_x_cache_](#)
- [CachedResults< SmartPtr< const Vector > > grad_kappa_times_damping_s_cache_](#)

Caches for constraint stuff

- [CachedResults< SmartPtr< const Vector > > curr_c_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_c_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_d_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_d_cache_](#)

- [CachedResults< SmartPtr< const Vector > > curr_d_minus_s_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_d_minus_s_cache_](#)
- [CachedResults< SmartPtr< const Matrix > > curr_jac_c_cache_](#)
- [CachedResults< SmartPtr< const Matrix > > trial_jac_c_cache_](#)
- [CachedResults< SmartPtr< const Matrix > > curr_jac_d_cache_](#)
- [CachedResults< SmartPtr< const Matrix > > trial_jac_d_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_jac_cT_times_vec_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_jac_cT_times_vec_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_jac_dT_times_vec_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_jac_dT_times_vec_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_jac_c_times_vec_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_jac_d_times_vec_cache_](#)
- [CachedResults< Number > curr_constraint_violation_cache_](#)
- [CachedResults< Number > trial_constraint_violation_cache_](#)
- [CachedResults< Number > curr_nlp_constraint_violation_cache_](#)
- [CachedResults< Number > unscaled_curr_nlp_constraint_violation_cache_](#)
- [CachedResults< Number > unscaled_trial_nlp_constraint_violation_cache_](#)

Components of primal-dual error

- [CachedResults< SmartPtr< const Vector > > curr_grad_lag_x_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_grad_lag_x_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_grad_lag_s_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_grad_lag_s_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_grad_lag_with_damping_x_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_grad_lag_with_damping_s_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_compl_x_L_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_compl_x_U_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_compl_s_L_cache_](#)
- [CachedResults< SmartPtr< const Vector > > curr_compl_s_U_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_compl_x_L_cache_](#)
- [CachedResults< SmartPtr< const Vector > > trial_compl_x_U_cache_](#)

- `CachedResults< SmartPtr< const Vector > > trial_compl_s_L_cache_`
- `CachedResults< SmartPtr< const Vector > > trial_compl_s_U_cache_`
- `CachedResults< SmartPtr< const Vector > > curr_relaxed_compl_x_L_cache_`
- `CachedResults< SmartPtr< const Vector > > curr_relaxed_compl_x_U_cache_`
- `CachedResults< SmartPtr< const Vector > > curr_relaxed_compl_s_L_cache_`
- `CachedResults< SmartPtr< const Vector > > curr_relaxed_compl_s_U_cache_`
- `CachedResults< Number > curr_primal_infeasibility_cache_`
- `CachedResults< Number > trial_primal_infeasibility_cache_`
- `CachedResults< Number > curr_dual_infeasibility_cache_`
- `CachedResults< Number > trial_dual_infeasibility_cache_`
- `CachedResults< Number > unscaled_curr_dual_infeasibility_cache_`
- `CachedResults< Number > curr_complementarity_cache_`
- `CachedResults< Number > trial_complementarity_cache_`
- `CachedResults< Number > curr_centrality_measure_cache_`
- `CachedResults< Number > curr_nlp_error_cache_`
- `CachedResults< Number > unscaled_curr_nlp_error_cache_`
- `CachedResults< Number > curr_barrier_error_cache_`
- `CachedResults< Number > curr_primal_dual_system_error_cache_`
- `CachedResults< Number > trial_primal_dual_system_error_cache_`

Caches for fraction to the boundary step sizes

- `CachedResults< Number > primal_frac_to_the_bound_cache_`
- `CachedResults< Number > dual_frac_to_the_bound_cache_`

Caches for sigma matrices

- `CachedResults< SmartPtr< const Vector > > curr_sigma_x_cache_`
- `CachedResults< SmartPtr< const Vector > > curr_sigma_s_cache_`

Indicator vectors required for the linear damping terms

to handle unbounded solution sets.

- `SmartPtr< Vector > dampind_x_L_`
Indicator vector for selecting the elements in x that have only lower bounds.
- `SmartPtr< Vector > dampind_x_U_`
Indicator vector for selecting the elements in x that have only upper bounds.
- `SmartPtr< Vector > dampind_s_L_`
Indicator vector for selecting the elements in s that have only lower bounds.
- `SmartPtr< Vector > dampind_s_U_`
Indicator vector for selecting the elements in s that have only upper bounds.

Temporary vectors for intermediate calculations. We keep

these around to avoid unnecessarily many new allocations of Vectors.

- `SmartPtr< Vector > tmp_x_`

- [SmartPointer< Vector > tmp_s_](#)
- [SmartPointer< Vector > tmp_c_](#)
- [SmartPointer< Vector > tmp_d_](#)
- [SmartPointer< Vector > tmp_x_L_](#)
- [SmartPointer< Vector > tmp_x_U_](#)
- [SmartPointer< Vector > tmp_s_L_](#)
- [SmartPointer< Vector > tmp_s_U_](#)
- [Vector & Tmp_x \(\)](#)

Accessor methods for the temporary vectors.

- [Vector & Tmp_s \(\)](#)
- [Vector & Tmp_c \(\)](#)
- [Vector & Tmp_d \(\)](#)
- [Vector & Tmp_x_L \(\)](#)
- [Vector & Tmp_x_U \(\)](#)
- [Vector & Tmp_s_L \(\)](#)
- [Vector & Tmp_s_U \(\)](#)

6.70.1 Detailed Description

Class for all IPOPT specific calculated quantities.

Definition at line 81 of file IpIpoptCalculatedQuantities.hpp.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 **Ipopt::IpoptCalculatedQuantities::IpoptCalculatedQuantities (const SmartPtr< IpoptNLP > & ip_nlp, const SmartPtr< IpoptData > & ip_data)**

Constructor.

6.70.2.2 **virtual Ipopt::IpoptCalculatedQuantities::~~IpoptCalculatedQuantities () [virtual]**

Default destructor.

6.70.2.3 **Ipopt::IpoptCalculatedQuantities::IpoptCalculatedQuantities () [private]**

Default Constructor.

6.70.2.4 **Ipopt::IpoptCalculatedQuantities::IpoptCalculatedQuantities (const IpoptCalculatedQuantities &) [private]**

Copy Constructor.

6.70.3 Member Function Documentation

6.70.3.1 **void Ipopt::IpoptCalculatedQuantities::SetAddCq (SmartPtr< IpoptAdditionalCq > add_cq) [inline]**

Method for setting pointer for additional calculated quantities.

This needs to be called before Initialized.

Definition at line 96 of file IpIpoptCalculatedQuantities.hpp.

6.70.3.2 **bool** Ipopt::IpoptCalculatedQuantities::HaveAddCq () [inline]

Method detecting if additional object for calculated quantities has already been set.

Definition at line 104 of file IpoptCalculatedQuantities.hpp.

6.70.3.3 **bool** Ipopt::IpoptCalculatedQuantities::Initialize (const **Journalist** & *jnlst*, const **OptionsList** & *options*, const std::string & *prefix*)

This method must be called to initialize the global algorithmic parameters.

The parameters are taken from the [OptionsList](#) object.

6.70.3.4 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::curr_slack_x_L ()

Slacks for x_L (at current iterate)

6.70.3.5 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::curr_slack_x_U ()

Slacks for x_U (at current iterate)

6.70.3.6 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::curr_slack_s_L ()

Slacks for s_L (at current iterate)

6.70.3.7 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::curr_slack_s_U ()

Slacks for s_U (at current iterate)

6.70.3.8 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::trial_slack_x_L ()

Slacks for x_L (at trial point)

6.70.3.9 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::trial_slack_x_U ()

Slacks for x_U (at trial point)

6.70.3.10 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::trial_slack_s_L ()

Slacks for s_L (at trial point)

6.70.3.11 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::trial_slack_s_U ()

Slacks for s_U (at trial point)

6.70.3.12 **Index** Ipopt::IpoptCalculatedQuantities::AdjustedTrialSlacks ()

Indicating whether or not we "fudged" the slacks.

6.70.3.13 **void** Ipopt::IpoptCalculatedQuantities::ResetAdjustedTrialSlacks ()

Reset the flags for "fudged" slacks.

6.70.3.14 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_f () [virtual]

Value of objective function (at current point)

6.70.3.15 **virtual Number** Ipopt::IpoptCalculatedQuantities::unscaled_curr_f() [virtual]

Unscaled value of the objective function (at the current point)

6.70.3.16 **virtual Number** Ipopt::IpoptCalculatedQuantities::trial_f() [virtual]

Value of objective function (at trial point)

6.70.3.17 **virtual Number** Ipopt::IpoptCalculatedQuantities::unscaled_trial_f() [virtual]

Unscaled value of the objective function (at the trial point)

6.70.3.18 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_grad_f()

Gradient of objective function (at current point)

6.70.3.19 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_grad_f()

Gradient of objective function (at trial point)

6.70.3.20 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_barrier_obj() [virtual]

Barrier Objective Function Value (at current iterate with current mu)

6.70.3.21 **virtual Number** Ipopt::IpoptCalculatedQuantities::trial_barrier_obj() [virtual]

Barrier Objective Function Value (at trial point with current mu)

6.70.3.22 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_grad_barrier_obj_x()

Gradient of barrier objective function with respect to x (at current point with current mu)

6.70.3.23 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_grad_barrier_obj_s()

Gradient of barrier objective function with respect to s (at current point with current mu)

6.70.3.24 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::grad_kappa_times_damping_x()

Gradient of the damping term with respect to x (times kappa_d)

6.70.3.25 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::grad_kappa_times_damping_s()

Gradient of the damping term with respect to s (times kappa_d)

6.70.3.26 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_c()

c(x) (at current point)

6.70.3.27 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::unscaled_curr_c()

unscaled c(x) (at current point)

6.70.3.28 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_c()

c(x) (at trial point)

6.70.3.29 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::unscaled_trial_c ()

unscaled $c(x)$ (at trial point)

6.70.3.30 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_d ()

$d(x)$ (at current point)

6.70.3.31 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::unscaled_curr_d ()

unscaled $d(x)$ (at current point)

6.70.3.32 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_d ()

$d(x)$ (at trial point)

6.70.3.33 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_d_minus_s ()

$d(x) - s$ (at current point)

6.70.3.34 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_d_minus_s ()

$d(x) - s$ (at trial point)

6.70.3.35 **SmartPtr<const Matrix>** Ipopt::IpoptCalculatedQuantities::curr_jac_c ()

Jacobian of c (at current point)

6.70.3.36 **SmartPtr<const Matrix>** Ipopt::IpoptCalculatedQuantities::trial_jac_c ()

Jacobian of c (at trial point)

6.70.3.37 **SmartPtr<const Matrix>** Ipopt::IpoptCalculatedQuantities::curr_jac_d ()

Jacobian of d (at current point)

6.70.3.38 **SmartPtr<const Matrix>** Ipopt::IpoptCalculatedQuantities::trial_jac_d ()

Jacobian of d (at trial point)

6.70.3.39 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_jac_cT_times_vec (const Vector & vec)

Product of Jacobian (evaluated at current point) of C transpose with general vector.

6.70.3.40 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_jac_cT_times_vec (const Vector & vec)

Product of Jacobian (evaluated at trial point) of C transpose with general vector.

6.70.3.41 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_jac_dT_times_vec (const Vector & vec)

Product of Jacobian (evaluated at current point) of D transpose with general vector.

6.70.3.42 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_jac_dT_times_vec (const Vector & vec)

Product of Jacobian (evaluated at trial point) of D transpose with general vector.

6.70.3.43 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_jac_cT_times_curr_y_c ()

Product of Jacobian (evaluated at current point) of C transpose with current y_c.

6.70.3.44 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_jac_cT_times_trial_y_c ()

Product of Jacobian (evaluated at trial point) of C transpose with trial y_c.

6.70.3.45 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_jac_dT_times_curr_y_d ()

Product of Jacobian (evaluated at current point) of D transpose with current y_d.

6.70.3.46 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_jac_dT_times_trial_y_d ()

Product of Jacobian (evaluated at trial point) of D transpose with trial y_d.

6.70.3.47 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_jac_c_times_vec (const Vector & vec)

Product of Jacobian (evaluated at current point) of C with general vector.

6.70.3.48 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_jac_d_times_vec (const Vector & vec)

Product of Jacobian (evaluated at current point) of D with general vector.

6.70.3.49 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_constraint_violation () [virtual]

Constraint Violation (at current iterate).

This value should be used in the line search, and not [curr_primal_infeasibility\(\)](#). What type of norm is used depends on constr_viol_normtype

6.70.3.50 **virtual Number** Ipopt::IpoptCalculatedQuantities::trial_constraint_violation () [virtual]

Constraint Violation (at trial point).

This value should be used in the line search, and not [curr_primal_infeasibility\(\)](#). What type of norm is used depends on constr_viol_normtype

6.70.3.51 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_nlp_constraint_violation (ENormType NormType) [virtual]

Real constraint violation in a given norm (at current iterate).

This considers the inequality constraints without slacks.

6.70.3.52 **virtual Number** Ipopt::IpoptCalculatedQuantities::unscaled_curr_nlp_constraint_violation (ENormType NormType) [virtual]

Unscaled real constraint violation in a given norm (at current iterate).

This considers the inequality constraints without slacks.

6.70.3.53 **virtual Number** Ipopt::IpoptCalculatedQuantities::unscaled_trial_nlp_constraint_violation (ENormType NormType) [virtual]

Unscaled real constraint violation in a given norm (at trial iterate).

This considers the inequality constraints without slacks.

6.70.3.54 **SmartPtr<const SymMatrix>** Ipopt::IpoptCalculatedQuantities::curr_exact_hessian ()

exact Hessian at current iterate (uncached)

6.70.3.55 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_grad_lag_x ()

x-part of gradient of Lagrangian function (at current point)

6.70.3.56 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_grad_lag_x ()

x-part of gradient of Lagrangian function (at trial point)

6.70.3.57 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_grad_lag_s ()

s-part of gradient of Lagrangian function (at current point)

6.70.3.58 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_grad_lag_s ()

s-part of gradient of Lagrangian function (at trial point)

6.70.3.59 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_grad_lag_with_damping_x ()

x-part of gradient of Lagrangian function (at current point)

including linear damping term

6.70.3.60 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_grad_lag_with_damping_s ()

s-part of gradient of Lagrangian function (at current point)

including linear damping term

6.70.3.61 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_compl_x_L ()

Complementarity for x_L (for current iterate)

6.70.3.62 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_compl_x_U ()

Complementarity for x_U (for current iterate)

6.70.3.63 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_compl_s_L ()

Complementarity for s_L (for current iterate)

6.70.3.64 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_compl_s_U ()

Complementarity for s_U (for current iterate)

6.70.3.65 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_compl_x_L ()

Complementarity for x_L (for trial iterate)

6.70.3.66 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_compl_x_U ()

Complementarity for x_U (for trial iterate)

6.70.3.67 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_compl_s_L ()

Complementarity for s_L (for trial iterate)

6.70.3.68 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::trial_compl_s_U ()

Complementarity for s_U (for trial iterate)

6.70.3.69 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_relaxed_compl_x_L ()

Relaxed complementarity for x_L (for current iterate and current mu)

6.70.3.70 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_relaxed_compl_x_U ()

Relaxed complementarity for x_U (for current iterate and current mu)

6.70.3.71 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_relaxed_compl_s_L ()

Relaxed complementarity for s_L (for current iterate and current mu)

6.70.3.72 **SmartPtr<const Vector>** Ipopt::IpoptCalculatedQuantities::curr_relaxed_compl_s_U ()

Relaxed complementarity for s_U (for current iterate and current mu)

6.70.3.73 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_primal_infeasibility (**ENormType NormType**) [virtual]

Primal infeasibility in a given norm (at current iterate).

6.70.3.74 **virtual Number** Ipopt::IpoptCalculatedQuantities::trial_primal_infeasibility (**ENormType NormType**) [virtual]

Primal infeasibility in a given norm (at trial point)

6.70.3.75 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_dual_infeasibility (**ENormType NormType**) [virtual]

Dual infeasibility in a given norm (at current iterate)

6.70.3.76 **virtual Number** Ipopt::IpoptCalculatedQuantities::trial_dual_infeasibility (**ENormType NormType**) [virtual]

Dual infeasibility in a given norm (at trial iterate)

6.70.3.77 **virtual Number** Ipopt::IpoptCalculatedQuantities::unscaled_curr_dual_infeasibility (**ENormType NormType**)
[virtual]

Unscaled dual infeasibility in a given norm (at current iterate)

6.70.3.78 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_complementarity (**Number mu**, **ENormType NormType**)
[virtual]

Complementarity (for all complementarity conditions together) in a given norm (at current iterate)

6.70.3.79 **virtual Number** Ipopt::IpoptCalculatedQuantities::trial_complementarity (**Number mu**, **ENormType NormType**)
[virtual]

Complementarity (for all complementarity conditions together) in a given norm (at trial iterate)

6.70.3.80 **virtual Number** Ipopt::IpoptCalculatedQuantities::unscaled_curr_complementarity (**Number** *mu*, **ENormType** *NormType*) [virtual]

Complementarity (for all complementarity conditions together) in a given norm (at current iterate) without NLP scaling.

6.70.3.81 **Number** Ipopt::IpoptCalculatedQuantities::CalcCentralityMeasure (const **Vector** & *compl_x_L*, const **Vector** & *compl_x_U*, const **Vector** & *compl_s_L*, const **Vector** & *compl_s_U*)

Centrality measure (in spirit of the -infinity-neighborhood).

6.70.3.82 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_centrality_measure () [virtual]

Centrality measure at current point.

6.70.3.83 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_nlp_error () [virtual]

Total optimality error for the original NLP at the current iterate, using scaling factors based on multipliers.

Note that here the constraint violation is measured without slacks (nlp_constraint_violation)

6.70.3.84 **virtual Number** Ipopt::IpoptCalculatedQuantities::unscaled_curr_nlp_error () [virtual]

Total optimality error for the original NLP at the current iterate, but using no scaling based on multipliers, and no scaling for the NLP.

Note that here the constraint violation is measured without slacks (nlp_constraint_violation)

6.70.3.85 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_barrier_error () [virtual]

Total optimality error for the barrier problem at the current iterate, using scaling factors based on multipliers.

6.70.3.86 **virtual Number** Ipopt::IpoptCalculatedQuantities::curr_primal_dual_system_error (**Number** *mu*) [virtual]

Norm of the primal-dual system for a given mu (at current iterate).

The norm is defined as the sum of the 1-norms of dual infeasibility, primal infeasibility, and complementarity, all divided by the number of elements of the vectors of which the norm is taken.

6.70.3.87 **virtual Number** Ipopt::IpoptCalculatedQuantities::trial_primal_dual_system_error (**Number** *mu*) [virtual]

Norm of the primal-dual system for a given mu (at trial iterate).

The norm is defined as the sum of the 1-norms of dual infeasibility, primal infeasibility, and complementarity, all divided by the number of elements of the vectors of which the norm is taken.

6.70.3.88 **Number** Ipopt::IpoptCalculatedQuantities::primal_frac_to_the_bound (**Number** *tau*, const **Vector** & *delta_x*, const **Vector** & *delta_s*)

Fraction to the boundary from (current) primal variables x and s for a given step.

6.70.3.89 **Number** Ipopt::IpoptCalculatedQuantities::curr_primal_frac_to_the_bound (**Number** *tau*)

Fraction to the boundary from (current) primal variables x and s for internal (current) step.

6.70.3.90 **Number** Ipopt::IpoptCalculatedQuantities::dual_frac_to_the_bound (**Number** *tau*, const **Vector** & *delta_z_L*, const **Vector** & *delta_z_U*, const **Vector** & *delta_v_L*, const **Vector** & *delta_v_U*)

Fraction to the boundary from (current) dual variables z and v for a given step.

6.70.3.91 **Number** Ipopt::IpoptCalculatedQuantities::uncached_dual_frac_to_the_bound (**Number** *tau*, const **Vector** & *delta_z_L*, const **Vector** & *delta_z_U*, const **Vector** & *delta_v_L*, const **Vector** & *delta_v_U*)

Fraction to the boundary from (current) dual variables z and v for a given step, without caching.

6.70.3.92 **Number** Ipopt::IpoptCalculatedQuantities::curr_dual_frac_to_the_bound (**Number** *tau*)

Fraction to the boundary from (current) dual variables z and v for internal (current) step.

6.70.3.93 **Number** Ipopt::IpoptCalculatedQuantities::uncached_slack_frac_to_the_bound (**Number** *tau*, const **Vector** & *delta_x_L*, const **Vector** & *delta_x_U*, const **Vector** & *delta_s_L*, const **Vector** & *delta_s_U*)

Fraction to the boundary from (current) slacks for a given step in the slacks.

Usually, one will use the `primal_frac_to_the_bound` method to compute the primal fraction to the boundary step size, but if it is cheaper to provide the steps in the slacks directly (e.g. when the primal step sizes are only temporary), the this method is more efficient. This method does not cache computations.

6.70.3.94 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::curr_sigma_x ()

6.70.3.95 **SmartPtr**<const **Vector**> Ipopt::IpoptCalculatedQuantities::curr_sigma_s ()

6.70.3.96 **Number** Ipopt::IpoptCalculatedQuantities::curr_avrg_compl ()

average of current values of the complementarities

6.70.3.97 **Number** Ipopt::IpoptCalculatedQuantities::trial_avrg_compl ()

average of trial values of the complementarities

6.70.3.98 **Number** Ipopt::IpoptCalculatedQuantities::curr_gradBarrTDelta ()

inner_product of current barrier obj.

fn. gradient with current search direction

6.70.3.99 **Number** Ipopt::IpoptCalculatedQuantities::CalcNormOfType (**ENormType** *NormType*, std::vector< **SmartPtr**<const **Vector**>> *vecs*)

Compute the norm of a specific type of a set of vectors (uncached)

6.70.3.100 **Number** Ipopt::IpoptCalculatedQuantities::CalcNormOfType (**ENormType** *NormType*, const **Vector** & *vec1*, const **Vector** & *vec2*)

Compute the norm of a specific type of two vectors (uncached)

6.70.3.101 **ENormType** Ipopt::IpoptCalculatedQuantities::constr_viol_normtype () const [inline]

Norm type used for calculating constraint violation.

Definition at line 437 of file `IpoptCalculatedQuantities.hpp`.

6.70.3.102 **bool** Ipopt::IpoptCalculatedQuantities::IsSquareProblem () const

Method returning true if this is a square problem.

6.70.3.103 **SmartPtr**<IpoptNLP> & Ipopt::IpoptCalculatedQuantities::GetIpoptNLP () [inline]

Method returning the `IpoptNLP` object.

This should only be used with care!

Definition at line 447 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.3.104 `IpoptAdditionalCq& Ipopt::IpoptCalculatedQuantities::AdditionalCq () [inline]`

Definition at line 452 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.3.105 `static void Ipopt::IpoptCalculatedQuantities::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for `IpoptType`.

Called by `IpoptType` to register the options

6.70.3.106 `void Ipopt::IpoptCalculatedQuantities::operator= (const IpoptCalculatedQuantities &) [private]`

Overloaded Equals Operator.

6.70.3.107 `Vector& Ipopt::IpoptCalculatedQuantities::Tmp_x () [private]`

Accessor methods for the temporary vectors.

6.70.3.108 `Vector& Ipopt::IpoptCalculatedQuantities::Tmp_s () [private]`

6.70.3.109 `Vector& Ipopt::IpoptCalculatedQuantities::Tmp_c () [private]`

6.70.3.110 `Vector& Ipopt::IpoptCalculatedQuantities::Tmp_d () [private]`

6.70.3.111 `Vector& Ipopt::IpoptCalculatedQuantities::Tmp_x_L () [private]`

6.70.3.112 `Vector& Ipopt::IpoptCalculatedQuantities::Tmp_x_U () [private]`

6.70.3.113 `Vector& Ipopt::IpoptCalculatedQuantities::Tmp_s_L () [private]`

6.70.3.114 `Vector& Ipopt::IpoptCalculatedQuantities::Tmp_s_U () [private]`

6.70.3.115 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::CalcSlack_L (const Matrix & P, const Vector & x, const Vector & x_bound) [private]`

Compute new vector containing the slack to a lower bound (uncached)

6.70.3.116 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::CalcSlack_U (const Matrix & P, const Vector & x, const Vector & x_bound) [private]`

Compute new vector containing the slack to a upper bound (uncached)

6.70.3.117 `Number Ipopt::IpoptCalculatedQuantities::CalcBarrierTerm (Number mu, const Vector & slack_x_L, const Vector & slack_x_U, const Vector & slack_s_L, const Vector & slack_s_U) [private]`

Compute barrier term at given point (uncached)

6.70.3.118 `SmartPtr<const Vector> Ipopt::IpoptCalculatedQuantities::CalcCompl (const Vector & slack, const Vector & mult) [private]`

Compute complementarity for slack / multiplier pair.

6.70.3.119 **Number** Ipopt::IpoptCalculatedQuantities::CalcFracToBound (const Vector & *slack_L*, Vector & *tmp_L*, const Matrix & *P_L*, const Vector & *slack_U*, Vector & *tmp_U*, const Matrix & *P_U*, const Vector & *delta*, Number *tau*) [private]

Compute fraction to the boundary parameter for lower and upper bounds.

6.70.3.120 **void** Ipopt::IpoptCalculatedQuantities::ComputeOptimalityErrorScaling (const Vector & *y_c*, const Vector & *y_d*, const Vector & *z_L*, const Vector & *z_U*, const Vector & *v_L*, const Vector & *v_U*, Number *s_max*, Number & *s_d*, Number & *s_c*) [private]

Compute the scaling factors for the optimality error.

6.70.3.121 **Index** Ipopt::IpoptCalculatedQuantities::CalculateSafeSlack (SmartPtr< Vector > & *slack*, const SmartPtr< const Vector > & *bound*, const SmartPtr< const Vector > & *curr_point*, const SmartPtr< const Vector > & *multiplier*) [private]

Check if slacks are becoming too small.

If slacks are becoming too small, they are change. The return value is the number of corrected slacks.

6.70.3.122 **void** Ipopt::IpoptCalculatedQuantities::ComputeDampingIndicators (SmartPtr< const Vector > & *dampind_x_L*, SmartPtr< const Vector > & *dampind_x_U*, SmartPtr< const Vector > & *dampind_s_L*, SmartPtr< const Vector > & *dampind_s_U*) [private]

Computes the indicator vectors that can be used to filter out those entries in the slack...

variables, that correspond to variables with only lower and upper bounds. This is required for the linear damping term in the barrier objective function to handle unbounded solution sets.

6.70.3.123 **bool** Ipopt::IpoptCalculatedQuantities::in_restoration_phase () [private]

Check if we are in the restoration phase.

Returns true, if the ip_nlp is of the type [RestIpoptNLP](#). ToDo: We probably want to handle this more elegant and don't have an explicit dependency here. Now I added this because otherwise the caching doesn't work properly since the restoration phase objective function depends on the current barrier parameter.

6.70.4 Member Data Documentation

6.70.4.1 **SmartPtr<IpoptNLP>** Ipopt::IpoptCalculatedQuantities::ip_nlp_ [private]

[Ipopt NLP](#) object.

Definition at line 486 of file IpoptCalculatedQuantities.hpp.

6.70.4.2 **SmartPtr<IpoptData>** Ipopt::IpoptCalculatedQuantities::ip_data_ [private]

[Ipopt Data](#) object.

Definition at line 488 of file IpoptCalculatedQuantities.hpp.

6.70.4.3 **SmartPtr<IpoptAdditionalCq>** Ipopt::IpoptCalculatedQuantities::add_cq_ [private]

Chen-Goldfarb specific calculated quantities.

Definition at line 490 of file IpoptCalculatedQuantities.hpp.

6.70.4.4 **Number** Ipopt::IpoptCalculatedQuantities::s_max_ [private]

Parameter in formula for computing overall primal-dual optimality error.

Definition at line 499 of file IpoptCalculatedQuantities.hpp.

6.70.4.5 **Number** Ipopt::IpoptCalculatedQuantities::kappa_d_ [private]

Weighting factor for the linear damping term added to the barrier objective function.

Definition at line 502 of file IpoptCalculatedQuantities.hpp.

6.70.4.6 **Number** Ipopt::IpoptCalculatedQuantities::slack_move_ [private]

fractional movement allowed in bounds

Definition at line 504 of file IpoptCalculatedQuantities.hpp.

6.70.4.7 **ENormType** Ipopt::IpoptCalculatedQuantities::constr_viol_normtype_ [private]

Norm type to be used when calculating the constraint violation.

Definition at line 506 of file IpoptCalculatedQuantities.hpp.

6.70.4.8 **bool** Ipopt::IpoptCalculatedQuantities::warm_start_same_structure_ [private]

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 509 of file IpoptCalculatedQuantities.hpp.

6.70.4.9 **Number** Ipopt::IpoptCalculatedQuantities::mu_target_ [private]

Desired value of the barrier parameter.

Definition at line 511 of file IpoptCalculatedQuantities.hpp.

6.70.4.10 **CachedResults< SmartPtr<Vector> >** Ipopt::IpoptCalculatedQuantities::curr_slack_x_L_cache_ [private]

Definition at line 516 of file IpoptCalculatedQuantities.hpp.

6.70.4.11 **CachedResults< SmartPtr<Vector> >** Ipopt::IpoptCalculatedQuantities::curr_slack_x_U_cache_ [private]

Definition at line 517 of file IpoptCalculatedQuantities.hpp.

6.70.4.12 **CachedResults< SmartPtr<Vector> >** Ipopt::IpoptCalculatedQuantities::curr_slack_s_L_cache_ [private]

Definition at line 518 of file IpoptCalculatedQuantities.hpp.

6.70.4.13 **CachedResults< SmartPtr<Vector> >** Ipopt::IpoptCalculatedQuantities::curr_slack_s_U_cache_ [private]

Definition at line 519 of file IpoptCalculatedQuantities.hpp.

6.70.4.14 **CachedResults< SmartPtr<Vector> >** Ipopt::IpoptCalculatedQuantities::trial_slack_x_L_cache_ [private]

Definition at line 520 of file IpoptCalculatedQuantities.hpp.

6.70.4.15 **CachedResults**< **SmartPtr**<**Vector**> > Ipopt::IpoptCalculatedQuantities::trial_slack_x_U_cache_ [private]

Definition at line 521 of file IpoptCalculatedQuantities.hpp.

6.70.4.16 **CachedResults**< **SmartPtr**<**Vector**> > Ipopt::IpoptCalculatedQuantities::trial_slack_s_L_cache_ [private]

Definition at line 522 of file IpoptCalculatedQuantities.hpp.

6.70.4.17 **CachedResults**< **SmartPtr**<**Vector**> > Ipopt::IpoptCalculatedQuantities::trial_slack_s_U_cache_ [private]

Definition at line 523 of file IpoptCalculatedQuantities.hpp.

6.70.4.18 **Index** Ipopt::IpoptCalculatedQuantities::num_adjusted_slack_x_L_ [private]

Definition at line 524 of file IpoptCalculatedQuantities.hpp.

6.70.4.19 **Index** Ipopt::IpoptCalculatedQuantities::num_adjusted_slack_x_U_ [private]

Definition at line 525 of file IpoptCalculatedQuantities.hpp.

6.70.4.20 **Index** Ipopt::IpoptCalculatedQuantities::num_adjusted_slack_s_L_ [private]

Definition at line 526 of file IpoptCalculatedQuantities.hpp.

6.70.4.21 **Index** Ipopt::IpoptCalculatedQuantities::num_adjusted_slack_s_U_ [private]

Definition at line 527 of file IpoptCalculatedQuantities.hpp.

6.70.4.22 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::curr_f_cache_ [private]

Definition at line 532 of file IpoptCalculatedQuantities.hpp.

6.70.4.23 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::trial_f_cache_ [private]

Definition at line 533 of file IpoptCalculatedQuantities.hpp.

6.70.4.24 **CachedResults**< **SmartPtr**<**const Vector**> > Ipopt::IpoptCalculatedQuantities::curr_grad_f_cache_
[private]

Definition at line 534 of file IpoptCalculatedQuantities.hpp.

6.70.4.25 **CachedResults**< **SmartPtr**<**const Vector**> > Ipopt::IpoptCalculatedQuantities::trial_grad_f_cache_
[private]

Definition at line 535 of file IpoptCalculatedQuantities.hpp.

6.70.4.26 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::curr_barrier_obj_cache_ [private]

Definition at line 540 of file IpoptCalculatedQuantities.hpp.

6.70.4.27 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::trial_barrier_obj_cache_ [private]

Definition at line 541 of file IpoptCalculatedQuantities.hpp.

6.70.4.28 **CachedResults**< **SmartPtr**<**const Vector**> > Ipopt::IpoptCalculatedQuantities::curr_grad_barrier_obj_x_cache_
[private]

Definition at line 542 of file IpoptCalculatedQuantities.hpp.

6.70.4.29 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_grad_barrier_obj_s_cache_`
[private]

Definition at line 543 of file lpoptCalculatedQuantities.hpp.

6.70.4.30 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::grad_kappa_times_damping_x_cache_`
[private]

Definition at line 544 of file lpoptCalculatedQuantities.hpp.

6.70.4.31 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::grad_kappa_times_damping_s_cache_`
[private]

Definition at line 545 of file lpoptCalculatedQuantities.hpp.

6.70.4.32 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_c_cache_` [private]

Definition at line 550 of file lpoptCalculatedQuantities.hpp.

6.70.4.33 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::trial_c_cache_` [private]

Definition at line 551 of file lpoptCalculatedQuantities.hpp.

6.70.4.34 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_d_cache_` [private]

Definition at line 552 of file lpoptCalculatedQuantities.hpp.

6.70.4.35 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::trial_d_cache_` [private]

Definition at line 553 of file lpoptCalculatedQuantities.hpp.

6.70.4.36 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_d_minus_s_cache_`
[private]

Definition at line 554 of file lpoptCalculatedQuantities.hpp.

6.70.4.37 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::trial_d_minus_s_cache_`
[private]

Definition at line 555 of file lpoptCalculatedQuantities.hpp.

6.70.4.38 `CachedResults< SmartPtr<const Matrix> > lpopt::lpoptCalculatedQuantities::curr_jac_c_cache_`
[private]

Definition at line 556 of file lpoptCalculatedQuantities.hpp.

6.70.4.39 `CachedResults< SmartPtr<const Matrix> > lpopt::lpoptCalculatedQuantities::trial_jac_c_cache_`
[private]

Definition at line 557 of file lpoptCalculatedQuantities.hpp.

6.70.4.40 `CachedResults< SmartPtr<const Matrix> > lpopt::lpoptCalculatedQuantities::curr_jac_d_cache_`
[private]

Definition at line 558 of file lpoptCalculatedQuantities.hpp.

6.70.4.41 **CachedResults**< **SmartPtr**<const **Matrix**> > Ipopt::IpoptCalculatedQuantities::trial_jac_d_cache_
[private]

Definition at line 559 of file IpoptCalculatedQuantities.hpp.

6.70.4.42 **CachedResults**< **SmartPtr**<const **Vector**> > Ipopt::IpoptCalculatedQuantities::curr_jac_cT_times_vec_cache_
[private]

Definition at line 560 of file IpoptCalculatedQuantities.hpp.

6.70.4.43 **CachedResults**< **SmartPtr**<const **Vector**> > Ipopt::IpoptCalculatedQuantities::trial_jac_cT_times_vec_cache_
[private]

Definition at line 561 of file IpoptCalculatedQuantities.hpp.

6.70.4.44 **CachedResults**< **SmartPtr**<const **Vector**> > Ipopt::IpoptCalculatedQuantities::curr_jac_dT_times_vec_cache_
[private]

Definition at line 562 of file IpoptCalculatedQuantities.hpp.

6.70.4.45 **CachedResults**< **SmartPtr**<const **Vector**> > Ipopt::IpoptCalculatedQuantities::trial_jac_dT_times_vec_cache_
[private]

Definition at line 563 of file IpoptCalculatedQuantities.hpp.

6.70.4.46 **CachedResults**< **SmartPtr**<const **Vector**> > Ipopt::IpoptCalculatedQuantities::curr_jac_c_times_vec_cache_
[private]

Definition at line 564 of file IpoptCalculatedQuantities.hpp.

6.70.4.47 **CachedResults**< **SmartPtr**<const **Vector**> > Ipopt::IpoptCalculatedQuantities::curr_jac_d_times_vec_cache_
[private]

Definition at line 565 of file IpoptCalculatedQuantities.hpp.

6.70.4.48 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::curr_constraint_violation_cache_ [private]

Definition at line 566 of file IpoptCalculatedQuantities.hpp.

6.70.4.49 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::trial_constraint_violation_cache_ [private]

Definition at line 567 of file IpoptCalculatedQuantities.hpp.

6.70.4.50 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::curr_nlp_constraint_violation_cache_ [private]

Definition at line 568 of file IpoptCalculatedQuantities.hpp.

6.70.4.51 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::unscaled_curr_nlp_constraint_violation_cache_
[private]

Definition at line 569 of file IpoptCalculatedQuantities.hpp.

6.70.4.52 **CachedResults**<**Number**> Ipopt::IpoptCalculatedQuantities::unscaled_trial_nlp_constraint_violation_cache_
[private]

Definition at line 570 of file IpoptCalculatedQuantities.hpp.

6.70.4.53 `CachedResults< SmartPtr<const SymMatrix> > lpopt::lpoptCalculatedQuantities::curr_exact_hessian_cache_`
`[private]`

Cache for the exact Hessian.

Definition at line 574 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.54 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_grad_lag_x_cache_`
`[private]`

Definition at line 578 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.55 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::trial_grad_lag_x_cache_`
`[private]`

Definition at line 579 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.56 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_grad_lag_s_cache_`
`[private]`

Definition at line 580 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.57 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::trial_grad_lag_s_cache_`
`[private]`

Definition at line 581 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.58 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_grad_lag_with_damping_x_ -`
`cache_ [private]`

Definition at line 582 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.59 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_grad_lag_with_damping_s_ -`
`cache_ [private]`

Definition at line 583 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.60 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_compl_x_L_cache_`
`[private]`

Definition at line 584 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.61 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_compl_x_U_cache_`
`[private]`

Definition at line 585 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.62 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_compl_s_L_cache_`
`[private]`

Definition at line 586 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.63 `CachedResults< SmartPtr<const Vector> > lpopt::lpoptCalculatedQuantities::curr_compl_s_U_cache_`
`[private]`

Definition at line 587 of file `lpoptCalculatedQuantities.hpp`.

6.70.4.64 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::trial_compl_x_L_cache_`
[private]

Definition at line 588 of file IpoptCalculatedQuantities.hpp.

6.70.4.65 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::trial_compl_x_U_cache_`
[private]

Definition at line 589 of file IpoptCalculatedQuantities.hpp.

6.70.4.66 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::trial_compl_s_L_cache_`
[private]

Definition at line 590 of file IpoptCalculatedQuantities.hpp.

6.70.4.67 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::trial_compl_s_U_cache_`
[private]

Definition at line 591 of file IpoptCalculatedQuantities.hpp.

6.70.4.68 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::curr_relaxed_compl_x_L_cache_`
[private]

Definition at line 592 of file IpoptCalculatedQuantities.hpp.

6.70.4.69 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::curr_relaxed_compl_x_U_cache_`
[private]

Definition at line 593 of file IpoptCalculatedQuantities.hpp.

6.70.4.70 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::curr_relaxed_compl_s_L_cache_`
[private]

Definition at line 594 of file IpoptCalculatedQuantities.hpp.

6.70.4.71 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::curr_relaxed_compl_s_U_cache_`
[private]

Definition at line 595 of file IpoptCalculatedQuantities.hpp.

6.70.4.72 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_primal_infeasibility_cache_` [private]

Definition at line 596 of file IpoptCalculatedQuantities.hpp.

6.70.4.73 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::trial_primal_infeasibility_cache_` [private]

Definition at line 597 of file IpoptCalculatedQuantities.hpp.

6.70.4.74 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_dual_infeasibility_cache_` [private]

Definition at line 598 of file IpoptCalculatedQuantities.hpp.

6.70.4.75 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::trial_dual_infeasibility_cache_` [private]

Definition at line 599 of file IpoptCalculatedQuantities.hpp.

6.70.4.76 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::unscaled_curr_dual_infeasibility_cache_`
`[private]`

Definition at line 600 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.77 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_complementarity_cache_` `[private]`

Definition at line 601 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.78 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::trial_complementarity_cache_` `[private]`

Definition at line 602 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.79 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_centrality_measure_cache_` `[private]`

Definition at line 603 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.80 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_nlp_error_cache_` `[private]`

Definition at line 604 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.81 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::unscaled_curr_nlp_error_cache_` `[private]`

Definition at line 605 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.82 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_barrier_error_cache_` `[private]`

Definition at line 606 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.83 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_primal_dual_system_error_cache_`
`[private]`

Definition at line 607 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.84 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::trial_primal_dual_system_error_cache_`
`[private]`

Definition at line 608 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.85 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::primal_frac_to_the_bound_cache_` `[private]`

Definition at line 613 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.86 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::dual_frac_to_the_bound_cache_` `[private]`

Definition at line 614 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.87 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::curr_sigma_x_cache_`
`[private]`

Definition at line 619 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.88 `CachedResults< SmartPtr<const Vector> > Ipopt::IpoptCalculatedQuantities::curr_sigma_s_cache_`
`[private]`

Definition at line 620 of file `IpIpoptCalculatedQuantities.hpp`.

6.70.4.89 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_avrg_compl_cache_ [private]`

Cache for average of current complementarity.

Definition at line 624 of file IpoptCalculatedQuantities.hpp.

6.70.4.90 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::trial_avrg_compl_cache_ [private]`

Cache for average of trial complementarity.

Definition at line 626 of file IpoptCalculatedQuantities.hpp.

6.70.4.91 `CachedResults<Number> Ipopt::IpoptCalculatedQuantities::curr_gradBarrTDelta_cache_ [private]`

Cache for grad barrier obj.

fn inner product with step

Definition at line 629 of file IpoptCalculatedQuantities.hpp.

6.70.4.92 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::dampind_x_L_ [private]`

Indicator vector for selecting the elements in x that have only lower bounds.

Definition at line 636 of file IpoptCalculatedQuantities.hpp.

6.70.4.93 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::dampind_x_U_ [private]`

Indicator vector for selecting the elements in x that have only upper bounds.

Definition at line 639 of file IpoptCalculatedQuantities.hpp.

6.70.4.94 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::dampind_s_L_ [private]`

Indicator vector for selecting the elements in s that have only lower bounds.

Definition at line 642 of file IpoptCalculatedQuantities.hpp.

6.70.4.95 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::dampind_s_U_ [private]`

Indicator vector for selecting the elements in s that have only upper bounds.

Definition at line 645 of file IpoptCalculatedQuantities.hpp.

6.70.4.96 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::tmp_x_ [private]`

Definition at line 652 of file IpoptCalculatedQuantities.hpp.

6.70.4.97 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::tmp_s_ [private]`

Definition at line 653 of file IpoptCalculatedQuantities.hpp.

6.70.4.98 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::tmp_c_ [private]`

Definition at line 654 of file IpoptCalculatedQuantities.hpp.

6.70.4.99 `SmartPtr<Vector> Ipopt::IpoptCalculatedQuantities::tmp_d_ [private]`

Definition at line 655 of file IpoptCalculatedQuantities.hpp.

6.70.4.100 **SmartPtr<Vector>** `Ipopt::IpoptCalculatedQuantities::tmp_x_L_` [private]

Definition at line 656 of file `IpoptCalculatedQuantities.hpp`.

6.70.4.101 **SmartPtr<Vector>** `Ipopt::IpoptCalculatedQuantities::tmp_x_U_` [private]

Definition at line 657 of file `IpoptCalculatedQuantities.hpp`.

6.70.4.102 **SmartPtr<Vector>** `Ipopt::IpoptCalculatedQuantities::tmp_s_L_` [private]

Definition at line 658 of file `IpoptCalculatedQuantities.hpp`.

6.70.4.103 **SmartPtr<Vector>** `Ipopt::IpoptCalculatedQuantities::tmp_s_U_` [private]

Definition at line 659 of file `IpoptCalculatedQuantities.hpp`.

6.70.4.104 **bool** `Ipopt::IpoptCalculatedQuantities::initialize_called_` [private]

flag indicating if Initialize method has been called (for debugging)

Definition at line 674 of file `IpoptCalculatedQuantities.hpp`.

The documentation for this class was generated from the following file:

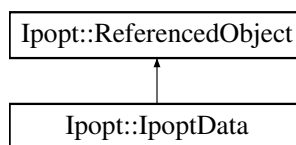
- [Algorithm/IpoptCalculatedQuantities.hpp](#)

6.71 Ipopt::IpoptData Class Reference

Class to organize all the data required by the algorithm.

```
#include <IpIpoptData.hpp>
```

Inheritance diagram for `Ipopt::IpoptData`:



Public Member Functions

- **bool** [InitializeDataStructures](#) ([IpoptNLP](#) &ip_nlp, bool want_x, bool want_y_c, bool want_y_d, bool want_z_L, bool want_z_U)
Initialize Data Structures.
- **bool** [Initialize](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)
This method must be called to initialize the global algorithmic parameters.
- **Number** [cpu_time_start](#) () const
Cpu time counter at the beginning of the optimization.
- [TimingStatistics](#) & [TimingStats](#) ()
Return Timing Statistics Object.
- **bool** [HaveAddData](#) ()
Check if additional data has been set.

- [IpoptAdditionalData](#) & [AdditionalData](#) ()
Get access to additional data object.
- void [SetAddData](#) ([SmartPtr](#)< [IpoptAdditionalData](#) > add_data)
Set a new pointer for additional [Ipopt](#) data.
- void [setPDPer](#) ([Number](#) pd_pert_x, [Number](#) pd_pert_s, [Number](#) pd_pert_c, [Number](#) pd_pert_d)
Set the perturbation of the primal-dual system.
- void [getPDPer](#) ([Number](#) &pd_pert_x, [Number](#) &pd_pert_s, [Number](#) &pd_pert_c, [Number](#) &pd_pert_d)
Get the current perturbation of the primal-dual system.

Constructors/Destructors

- [IpoptData](#) ([SmartPtr](#)< [IpoptAdditionalData](#) > add_data=NULL, [Number](#) cpu_time_start=-1.)
Constructor.
- virtual [~IpoptData](#) ()
Default destructor.

Get Methods for Iterates

- [SmartPtr](#)< const [IteratesVector](#) > [curr](#) () const
Current point.
- [SmartPtr](#)< const [IteratesVector](#) > [trial](#) () const
Get the current point in a copied container that is non-const.
- void [set_trial](#) ([SmartPtr](#)< [IteratesVector](#) > &trial)
Get Trial point in a copied container that is non-const.
- void [SetTrialPrimalVariablesFromStep](#) ([Number](#) alpha, const [Vector](#) &delta_x, const [Vector](#) &delta_s)
Set the values of the primal trial variables (x and s) from provided Step with step length alpha.
- void [SetTrialEqMultipliersFromStep](#) ([Number](#) alpha, const [Vector](#) &delta_y_c, const [Vector](#) &delta_y_d)
Set the values of the trial values for the equality constraint multipliers (y_c and y_d) from provided step with step length alpha.
- void [SetTrialBoundMultipliersFromStep](#) ([Number](#) alpha, const [Vector](#) &delta_z_L, const [Vector](#) &delta_z_U, const [Vector](#) &delta_v_L, const [Vector](#) &delta_v_U)
Set the value of the trial values for the bound multipliers (z_L, z_U, v_L, v_U) from provided step with step length alpha.
- [SmartPtr](#)< const [IteratesVector](#) > [delta](#) () const
ToDo: I may need to add versions of [set_trial](#) like the following, but I am not sure.
- void [set_delta](#) ([SmartPtr](#)< [IteratesVector](#) > &delta)
Set the current delta - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.
- void [set_delta](#) ([SmartPtr](#)< const [IteratesVector](#) > &delta)
Set the current delta - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.
- [SmartPtr](#)< const [IteratesVector](#) > [delta_aff](#) () const
Affine Delta.
- void [set_delta_aff](#) ([SmartPtr](#)< [IteratesVector](#) > &delta_aff)
Set the affine delta - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.
- [SmartPtr](#)< const [SymMatrix](#) > [W](#) ()
Hessian or Hessian approximation (do not hold on to it, it might be changed)
- void [Set_W](#) ([SmartPtr](#)< const [SymMatrix](#) > W)
Set Hessian approximation.

("Main") Primal-dual search direction. Those fields are

used to store the search directions computed from solving the primal-dual system, and can be used in the line search.

They are overwritten in every iteration, so do not hold on to the pointers (make copies instead)

- bool [HaveDeltas](#) () const
Returns true, if the primal-dual step have been already computed for the current iteration.
- void [SetHaveDeltas](#) (bool have_deltas)
Method for setting the HaveDeltas flag.

Affine-scaling step. Those fields can be used to store

the affine scaling step.

For example, if the method for computing the current barrier parameter computes the affine scaling steps, then the corrector step in the line search does not have to recompute those solutions of the linear system.

- bool [HaveAffineDeltas](#) () const
Returns true, if the affine-scaling step have been already computed for the current iteration.
- void [SetHaveAffineDeltas](#) (bool have_affine_deltas)
Method for setting the HaveDeltas flag.

Public Methods for updating iterates

- void [CopyTrialToCurrent](#) ()
Copy the trial values to the current values.
- void [AcceptTrialPoint](#) ()
Set the current iterate values from the trial values.

General algorithmic data

- [Index iter_count](#) () const
- void [Set_iter_count](#) (Index iter_count)
- [Number curr_mu](#) () const
- void [Set_mu](#) (Number mu)
- bool [MuInitialized](#) () const
- [Number curr_tau](#) () const
- void [Set_tau](#) (Number tau)
- bool [TauInitialized](#) () const
- void [SetFreeMuMode](#) (bool free_mu_mode)
- bool [FreeMuMode](#) () const
- void [Set_tiny_step_flag](#) (bool flag)
Setting the flag that indicates if a tiny step (below machine precision) has been detected.
- bool [tiny_step_flag](#) ()
- [Number tol](#) () const
Overall convergence tolerance.
- void [Set_tol](#) (Number tol)
Set a new value for the tolerance.

Information gathered for iteration output

- [Number info_regu_x](#) () const
- void [Set_info_regu_x](#) (Number regu_x)
- [Number info_alpha_primal](#) () const
- void [Set_info_alpha_primal](#) (Number alpha_primal)
- char [info_alpha_primal_char](#) () const
- void [Set_info_alpha_primal_char](#) (char info_alpha_primal_char)
- [Number info_alpha_dual](#) () const
- void [Set_info_alpha_dual](#) (Number alpha_dual)
- [Index info_ls_count](#) () const

- void [Set_info_ls_count](#) ([Index](#) ls_count)
- bool [info_skip_output](#) () const
- void [Append_info_string](#) (const std::string &add_str)
- const std::string & [info_string](#) () const
- void [Set_info_skip_output](#) (bool [info_skip_output](#))
Set this to true, if the next time when output is written, the summary line should not be printed.
- [Number](#) [info_last_output](#) ()
gives time when the last summary output line was printed
- void [Set_info_last_output](#) ([Number](#) [info_last_output](#))
sets time when the last summary output line was printed
- int [info_iters_since_header](#) ()
gives number of iteration summaries actually printed since last summary header was printed
- void [Inc_info_iters_since_header](#) ()
increases number of iteration summaries actually printed since last summary header was printed
- void [Set_info_iters_since_header](#) (int [info_iters_since_header](#))
sets number of iteration summaries actually printed since last summary header was printed
- void [ResetInfo](#) ()
Reset all info fields.

Static Public Member Functions

- static void [RegisterOptions](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IpoptData](#) (const [IpoptData](#) &)
Copy Constructor.
- void [operator=](#) (const [IpoptData](#) &)
Overloaded Equals Operator.

Private Attributes

- [Index](#) [iter_count_](#)
iteration count
- [Number](#) [curr_mu_](#)
current barrier parameter
- bool [mu_initialized_](#)
- [Number](#) [curr_tau_](#)
current fraction to the boundary parameter
- bool [tau_initialized_](#)
- bool [initialize_called_](#)
flag indicating if Initialize method has been called (for debugging)
- bool [have_prototypes_](#)
flag for debugging whether we have already curr_ values available (from which new Vectors can be generated)

- [SmartPtr< IteratesVectorSpace > iterates_space_](#)
VectorSpace for all the iterates.
- [TimingStatistics timing_statistics_](#)
TimingStatistics object collecting all [lpopt](#) timing statistics.
- [Number cpu_time_start_](#)
CPU time counter at initialization.
- [SmartPtr< lpoptAdditionalData > add_data_](#)
Object for the data specific for the Chen-Goldfarb penalty method algorithm.

Iterates

- [SmartPtr< const IteratesVector > curr_](#)
Main iteration variables (current iteration)
- [SmartPtr< const IteratesVector > trial_](#)
Main iteration variables (trial calculations)
- [SmartPtr< const SymMatrix > W_](#)
Hessian (approximation) - might be changed elsewhere!

Primal-dual Step

- [SmartPtr< const IteratesVector > delta_](#)
- [bool have_deltas_](#)
The following flag is set to true, if some other part of the algorithm (like the method for computing the barrier parameter) has already computed the primal-dual search direction.

Affine-scaling step. This used to transfer the

information about the affine-scaling step from the computation of the barrier parameter to the corrector (in the line search).

- [SmartPtr< const IteratesVector > delta_aff_](#)
- [bool have_affine_deltas_](#)
The following flag is set to true, if some other part of the algorithm (like the method for computing the barrier parameter) has already computed the affine-scaling step.

Global algorithm parameters. Those are options that can

be modified by the user and appear at different places in the algorithm.

They are set using an [OptionsList](#) object in the *Initialize* method.

- [Number tol_](#)
Overall convergence tolerance.

Status data

- [bool free_mu_mode_](#)
flag indicating whether the algorithm is in the free mu mode
- [bool tiny_step_flag_](#)
flag indicating if a tiny step has been detected

Gathered information for iteration output

- [Number info_regu_x_](#)
Size of regularization for the Hessian.

- [Number info_alpha_primal_](#)
Primal step size.
- [char info_alpha_primal_char_](#)
Info character for primal step size.
- [Number info_alpha_dual_](#)
Dual step size.
- [Index info_ls_count_](#)
Number of backtracking trial steps.
- [bool info_skip_output_](#)
true, if next summary output line should not be printed (eg after restoration phase).
- [std::string info_string_](#)
any string of characters for the end of the output line
- [Number info_last_output_](#)
time when the last summary output line was printed
- [int info_iters_since_header_](#)
number of iteration summaries actually printed since last summary header was printed

Information about the perturbation of the primal-dual

system

- [Number pd_pert_x_](#)
- [Number pd_pert_s_](#)
- [Number pd_pert_c_](#)
- [Number pd_pert_d_](#)

6.71.1 Detailed Description

Class to organize all the data required by the algorithm.

Internally, once this Data object has been initialized, all internal curr_ vectors must always be set (so that prototypes are available). The current values can only be set from the trial values. The trial values can be set by copying from a vector or by adding some fraction of a step to the current values. This object also stores steps, which allows to easily communicate the step from the step computation object to the line search object.

Definition at line 83 of file IpoptData.hpp.

6.71.2 Constructor & Destructor Documentation

6.71.2.1 `Ipopt::IpoptData::IpoptData (SmartPtr< IpoptAdditionalData > add_data = NULL, Number cpu_time_start = -1 .)`

Constructor.

6.71.2.2 `virtual Ipopt::IpoptData::~IpoptData () [virtual]`

Default destructor.

6.71.2.3 `Ipopt::IpoptData::IpoptData (const IpoptData &) [private]`

Copy Constructor.

6.71.3 Member Function Documentation

6.71.3.1 `bool lpopt::lpoptData::InitializeDataStructures (lpoptNLP & ip_nlp, bool want_x, bool want_y_c, bool want_y_d, bool want_z_L, bool want_z_U)`

Initialize Data Structures.

6.71.3.2 `bool lpopt::lpoptData::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix)`

This method must be called to initialize the global algorithmic parameters.

The parameters are taken from the [OptionsList](#) object.

6.71.3.3 `SmartPtr< const IteratesVector > lpopt::lpoptData::curr () const [inline]`

Current point.

Definition at line 690 of file `lpoptData.hpp`.

6.71.3.4 `SmartPtr< const IteratesVector > lpopt::lpoptData::trial () const [inline]`

Get the current point in a copied container that is non-const.

The entries in the container cannot be modified, but the container can be modified to point to new entries. Get Trial point

Definition at line 698 of file `lpoptData.hpp`.

6.71.3.5 `void lpopt::lpoptData::set_trial (SmartPtr< IteratesVector > & trial) [inline]`

Get Trial point in a copied container that is non-const.

The entries in the container can not be modified, but the container can be modified to point to new entries. Set the trial point - this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set you cannot modify the data again

Definition at line 740 of file `lpoptData.hpp`.

6.71.3.6 `void lpopt::lpoptData::SetTrialPrimalVariablesFromStep (Number alpha, const Vector & delta_x, const Vector & delta_s)`

Set the values of the primal trial variables (x and s) from provided Step with step length alpha.

6.71.3.7 `void lpopt::lpoptData::SetTrialEqMultipliersFromStep (Number alpha, const Vector & delta_y_c, const Vector & delta_y_d)`

Set the values of the trial values for the equality constraint multipliers (y_c and y_d) from provided step with step length alpha.

6.71.3.8 `void lpopt::lpoptData::SetTrialBoundMultipliersFromStep (Number alpha, const Vector & delta_z_L, const Vector & delta_z_U, const Vector & delta_v_L, const Vector & delta_v_U)`

Set the value of the trial values for the bound multipliers (z_L, z_U, v_L, v_U) from provided step with step length alpha.

6.71.3.9 `SmartPtr< const IteratesVector > lpopt::lpoptData::delta () const [inline]`

ToDo: I may need to add versions of set_trial like the following, but I am not sure.

get the current delta

Definition at line 706 of file `lpoptData.hpp`.

6.71.3.10 `void Ipopt::IpoptData::set_delta (SmartPtr< IteratesVector > & delta) [inline]`

Set the current delta - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.

Definition at line 761 of file IpoptData.hpp.

6.71.3.11 `void Ipopt::IpoptData::set_delta (SmartPtr< const IteratesVector > & delta) [inline]`

Set the current delta - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.

This is the version that is happy with a pointer to const [IteratesVector](#).

Definition at line 780 of file IpoptData.hpp.

6.71.3.12 `SmartPtr< const IteratesVector > Ipopt::IpoptData::delta_aff () const [inline]`

Affine Delta.

Definition at line 714 of file IpoptData.hpp.

6.71.3.13 `void Ipopt::IpoptData::set_delta_aff (SmartPtr< IteratesVector > & delta_aff) [inline]`

Set the affine delta - like the trial point, this method copies the pointer for efficiency (no copy and to keep cache tags the same) so after you call set, you cannot modify the data.

Definition at line 799 of file IpoptData.hpp.

6.71.3.14 `SmartPtr<const SymMatrix> Ipopt::IpoptData::W () [inline]`

Hessian or Hessian approximation (do not hold on to it, it might be changed)

Definition at line 201 of file IpoptData.hpp.

6.71.3.15 `void Ipopt::IpoptData::Set_W (SmartPtr< const SymMatrix > W) [inline]`

Set Hessian approximation.

Definition at line 208 of file IpoptData.hpp.

6.71.3.16 `bool Ipopt::IpoptData::HaveDeltas () const [inline]`

Returns true, if the primal-dual step have been already computed for the current iteration.

This flag is reset after every call of [AcceptTrialPoint\(\)](#). If the search direction is computed during the computation of the barrier parameter, the method computing the barrier parameter should call [SetHaveDeltas\(true\)](#) to tell the [IpoptAlgorithm](#) object that it doesn't need to recompute the primal-dual step.

Definition at line 227 of file IpoptData.hpp.

6.71.3.17 `void Ipopt::IpoptData::SetHaveDeltas (bool have_deltas) [inline]`

Method for setting the HaveDeltas flag.

This method should be called if some method computes the primal-dual step (and stores it in the delta_ fields of [IpoptData](#)) at an early part of the iteration. If that flag is set to true, the [IpoptAlgorithm](#) object will not recompute the step.

Definition at line 237 of file IpoptData.hpp.

6.71.3.18 `bool Ipopt::IpoptData::HaveAffineDeltas () const [inline]`

Returns true, if the affine-scaling step have been already computed for the current iteration.

This flag is reset after every call of [AcceptTrialPoint\(\)](#). If the search direction is computed during the computation of the barrier parameter, the method computing the barrier parameter should call `SetHaveDeltas(true)` to tell the line search does not have to recompute them in case it wants to do a corrector step.

Definition at line 257 of file `IpoptData.hpp`.

6.71.3.19 `void Ipopt::IpoptData::SetHaveAffineDeltas (bool have_affine_deltas) [inline]`

Method for setting the HaveDeltas flag.

This method should be called if some method computes the primal-dual step (and stores it in the `delta_` fields of [IpoptData](#)) at an early part of the iteration. If that flag is set to true, the [IpoptAlgorithm](#) object will not recompute the step.

Definition at line 267 of file `IpoptData.hpp`.

6.71.3.20 `void Ipopt::IpoptData::CopyTrialToCurrent () [inline]`

Copy the trial values to the current values.

Definition at line 722 of file `IpoptData.hpp`.

6.71.3.21 `void Ipopt::IpoptData::AcceptTrialPoint ()`

Set the current iterate values from the trial values.

6.71.3.22 `Index Ipopt::IpoptData::iter_count () const [inline]`

Definition at line 286 of file `IpoptData.hpp`.

6.71.3.23 `void Ipopt::IpoptData::Set_iter_count (Index iter_count) [inline]`

Definition at line 290 of file `IpoptData.hpp`.

6.71.3.24 `Number Ipopt::IpoptData::curr_mu () const [inline]`

Definition at line 295 of file `IpoptData.hpp`.

6.71.3.25 `void Ipopt::IpoptData::Set_mu (Number mu) [inline]`

Definition at line 300 of file `IpoptData.hpp`.

6.71.3.26 `bool Ipopt::IpoptData::Mulinitialized () const [inline]`

Definition at line 305 of file `IpoptData.hpp`.

6.71.3.27 `Number Ipopt::IpoptData::curr_tau () const [inline]`

Definition at line 310 of file `IpoptData.hpp`.

6.71.3.28 `void Ipopt::IpoptData::Set_tau (Number tau) [inline]`

Definition at line 315 of file `IpoptData.hpp`.

6.71.3.29 `bool Ipopt::IpoptData::Tauinitialized () const [inline]`

Definition at line 320 of file `IpoptData.hpp`.

6.71.3.30 `void Ipopt::IpoptData::SetFreeMuMode (bool free_mu_mode) [inline]`

Definition at line 325 of file `IpoptData.hpp`.

6.71.3.31 `bool Ipopt::IpoptData::FreeMuMode () const [inline]`

Definition at line 329 of file `IpoptData.hpp`.

6.71.3.32 `void Ipopt::IpoptData::Set_tiny_step_flag (bool flag) [inline]`

Setting the flag that indicates if a tiny step (below machine precision) has been detected.

Definition at line 336 of file `IpoptData.hpp`.

6.71.3.33 `bool Ipopt::IpoptData::tiny_step_flag () [inline]`

Definition at line 340 of file `IpoptData.hpp`.

6.71.3.34 `Number Ipopt::IpoptData::tol () const [inline]`

Overall convergence tolerance.

It is used in the convergence test, but also in some other parts of the algorithm that depend on the specified tolerance, such as the minimum value for the barrier parameter. Obtain the tolerance.

Definition at line 352 of file `IpoptData.hpp`.

6.71.3.35 `void Ipopt::IpoptData::Set_tol (Number tol) [inline]`

Set a new value for the tolerance.

One should be very careful when using this, since changing the predefined tolerance might have unexpected consequences. This method is for example used in the restoration convergence checker to tighten the restoration phase convergence tolerance, if the restoration phase converged to a point that has not a large value for the constraint violation.

Definition at line 364 of file `IpoptData.hpp`.

6.71.3.36 `Number Ipopt::IpoptData::cpu_time_start () const [inline]`

Cpu time counter at the beginning of the optimization.

This is useful to see how much CPU time has been spent in this optimization run.

Definition at line 373 of file `IpoptData.hpp`.

6.71.3.37 `Number Ipopt::IpoptData::info_regu_x () const [inline]`

Definition at line 380 of file `IpoptData.hpp`.

6.71.3.38 `void Ipopt::IpoptData::Set_info_regu_x (Number regu_x) [inline]`

Definition at line 384 of file `IpoptData.hpp`.

6.71.3.39 `Number Ipopt::IpoptData::info_alpha_primal () const [inline]`

Definition at line 388 of file `IpoptData.hpp`.

6.71.3.40 `void Ipopt::IpoptData::Set_info_alpha_primal (Number alpha_primal) [inline]`

Definition at line 392 of file `IpoptData.hpp`.

6.71.3.41 `char lpopt::lpoptData::info_alpha_primal_char () const [inline]`

Definition at line 396 of file lpoptData.hpp.

6.71.3.42 `void lpopt::lpoptData::Set_info_alpha_primal_char (char info_alpha_primal_char) [inline]`

Definition at line 400 of file lpoptData.hpp.

6.71.3.43 `Number lpopt::lpoptData::info_alpha_dual () const [inline]`

Definition at line 404 of file lpoptData.hpp.

6.71.3.44 `void lpopt::lpoptData::Set_info_alpha_dual (Number alpha_dual) [inline]`

Definition at line 408 of file lpoptData.hpp.

6.71.3.45 `Index lpopt::lpoptData::info_ls_count () const [inline]`

Definition at line 412 of file lpoptData.hpp.

6.71.3.46 `void lpopt::lpoptData::Set_info_ls_count (Index ls_count) [inline]`

Definition at line 416 of file lpoptData.hpp.

6.71.3.47 `bool lpopt::lpoptData::info_skip_output () const [inline]`

Definition at line 420 of file lpoptData.hpp.

6.71.3.48 `void lpopt::lpoptData::Append_info_string (const std::string & add_str) [inline]`

Definition at line 424 of file lpoptData.hpp.

6.71.3.49 `const std::string& lpopt::lpoptData::info_string () const [inline]`

Definition at line 428 of file lpoptData.hpp.

6.71.3.50 `void lpopt::lpoptData::Set_info_skip_output (bool info_skip_output) [inline]`

Set this to true, if the next time when output is written, the summary line should not be printed.

Definition at line 434 of file lpoptData.hpp.

6.71.3.51 `Number lpopt::lpoptData::info_last_output () [inline]`

gives time when the last summary output line was printed

Definition at line 440 of file lpoptData.hpp.

6.71.3.52 `void lpopt::lpoptData::Set_info_last_output (Number info_last_output) [inline]`

sets time when the last summary output line was printed

Definition at line 445 of file lpoptData.hpp.

6.71.3.53 `int lpopt::lpoptData::info_iters_since_header () [inline]`

gives number of iteration summaries actually printed since last summary header was printed

Definition at line 452 of file lpoptData.hpp.

6.71.3.54 void Ipopt::IpoptData::Inc_info_iters_since_header () [inline]

increases number of iteration summaries actually printed since last summary header was printed

Definition at line 458 of file IpoptData.hpp.

6.71.3.55 void Ipopt::IpoptData::Set_info_iters_since_header (int *info_iters_since_header*) [inline]

sets number of iteration summaries actually printed since last summary header was printed

Definition at line 464 of file IpoptData.hpp.

6.71.3.56 void Ipopt::IpoptData::ResetInfo () [inline]

Reset all info fields.

Definition at line 470 of file IpoptData.hpp.

6.71.3.57 TimingStatistics& Ipopt::IpoptData::TimingStats () [inline]

Return Timing Statistics Object.

Definition at line 482 of file IpoptData.hpp.

6.71.3.58 bool Ipopt::IpoptData::HaveAddData () [inline]

Check if additional data has been set.

Definition at line 488 of file IpoptData.hpp.

6.71.3.59 IpoptAdditionalData& Ipopt::IpoptData::AdditionalData () [inline]

Get access to additional data object.

Definition at line 494 of file IpoptData.hpp.

6.71.3.60 void Ipopt::IpoptData::SetAddData (SmartPtr< IpoptAdditionalData > *add_data*) [inline]

Set a new pointer for additional [Ipopt](#) data.

Definition at line 500 of file IpoptData.hpp.

6.71.3.61 void Ipopt::IpoptData::setPDPert (Number *pd_pert_x*, Number *pd_pert_s*, Number *pd_pert_c*, Number *pd_pert_d*) [inline]

Set the perturbation of the primal-dual system.

Definition at line 507 of file IpoptData.hpp.

6.71.3.62 void Ipopt::IpoptData::getPDPert (Number & *pd_pert_x*, Number & *pd_pert_s*, Number & *pd_pert_c*, Number & *pd_pert_d*) [inline]

Get the current perturbation of the primal-dual system.

Definition at line 517 of file IpoptData.hpp.

6.71.3.63 static void Ipopt::IpoptData::RegisterOptions (const SmartPtr< RegisteredOptions > & *options*) [static]

Methods for IpoptType.

6.71.3.64 `void Ipopt::IpoptData::operator= (const IpoptData &) [private]`

Overloaded Equals Operator.

6.71.4 Member Data Documentation

6.71.4.1 `SmartPtr<const IteratesVector> Ipopt::IpoptData::curr_ [private]`

Main iteration variables (current iteration)

Definition at line 536 of file IpoptData.hpp.

6.71.4.2 `SmartPtr<const IteratesVector> Ipopt::IpoptData::trial_ [private]`

Main iteration variables (trial calculations)

Definition at line 540 of file IpoptData.hpp.

6.71.4.3 `SmartPtr<const SymMatrix> Ipopt::IpoptData::W_ [private]`

Hessian (approximation) - might be changed elsewhere!

Definition at line 543 of file IpoptData.hpp.

6.71.4.4 `SmartPtr<const IteratesVector> Ipopt::IpoptData::delta_ [private]`

Definition at line 547 of file IpoptData.hpp.

6.71.4.5 `bool Ipopt::IpoptData::have_deltas_ [private]`

The following flag is set to true, if some other part of the algorithm (like the method for computing the barrier parameter) has already computed the primal-dual search direction.

This flag is reset when the AcceptTrialPoint method is called. To Do: we could cue off of a null delta_;

Definition at line 555 of file IpoptData.hpp.

6.71.4.6 `SmartPtr<const IteratesVector> Ipopt::IpoptData::delta_aff_ [private]`

Definition at line 563 of file IpoptData.hpp.

6.71.4.7 `bool Ipopt::IpoptData::have_affine_deltas_ [private]`

The following flag is set to true, if some other part of the algorithm (like the method for computing the barrier parameter) has already computed the affine-scaling step.

This flag is reset when the AcceptTrialPoint method is called. To Do: we could cue off of a null delta_aff_;

Definition at line 570 of file IpoptData.hpp.

6.71.4.8 `Index Ipopt::IpoptData::iter_count_ [private]`

iteration count

Definition at line 574 of file IpoptData.hpp.

6.71.4.9 `Number Ipopt::IpoptData::curr_mu_ [private]`

current barrier parameter

Definition at line 577 of file IpoptData.hpp.

6.71.4.10 `bool Ipopt::IpoptData::mu_initialized_ [private]`

Definition at line 578 of file IpoptData.hpp.

6.71.4.11 `Number Ipopt::IpoptData::curr_tau_ [private]`

current fraction to the boundary parameter

Definition at line 581 of file IpoptData.hpp.

6.71.4.12 `bool Ipopt::IpoptData::tau_initialized_ [private]`

Definition at line 582 of file IpoptData.hpp.

6.71.4.13 `bool Ipopt::IpoptData::initialize_called_ [private]`

flag indicating if Initialize method has been called (for debugging)

Definition at line 586 of file IpoptData.hpp.

6.71.4.14 `bool Ipopt::IpoptData::have_prototypes_ [private]`

flag for debugging whether we have already curr_ values available (from which new Vectors can be generated)

Definition at line 590 of file IpoptData.hpp.

6.71.4.15 `Number Ipopt::IpoptData::tol_ [private]`

Overall convergence tolerance.

Definition at line 598 of file IpoptData.hpp.

6.71.4.16 `bool Ipopt::IpoptData::free_mu_mode_ [private]`

flag indicating whether the algorithm is in the free mu mode

Definition at line 604 of file IpoptData.hpp.

6.71.4.17 `bool Ipopt::IpoptData::tiny_step_flag_ [private]`

flag indicating if a tiny step has been detected

Definition at line 606 of file IpoptData.hpp.

6.71.4.18 `Number Ipopt::IpoptData::info_regu_x_ [private]`

Size of regularization for the Hessian.

Definition at line 612 of file IpoptData.hpp.

6.71.4.19 `Number Ipopt::IpoptData::info_alpha_primal_ [private]`

Primal step size.

Definition at line 614 of file IpoptData.hpp.

6.71.4.20 `char Ipopt::IpoptData::info_alpha_primal_char_ [private]`

Info character for primal step size.

Definition at line 616 of file `lpoptData.hpp`.

6.71.4.21 `Number lpopt::lpoptData::info_alpha_dual_ [private]`

Dual step size.

Definition at line 618 of file `lpoptData.hpp`.

6.71.4.22 `Index lpopt::lpoptData::info_ls_count_ [private]`

Number of backtracking trial steps.

Definition at line 620 of file `lpoptData.hpp`.

6.71.4.23 `bool lpopt::lpoptData::info_skip_output_ [private]`

true, if next summary output line should not be printed (eg after restoration phase).

Definition at line 623 of file `lpoptData.hpp`.

6.71.4.24 `std::string lpopt::lpoptData::info_string_ [private]`

any string of characters for the end of the output line

Definition at line 625 of file `lpoptData.hpp`.

6.71.4.25 `Number lpopt::lpoptData::info_last_output_ [private]`

time when the last summary output line was printed

Definition at line 627 of file `lpoptData.hpp`.

6.71.4.26 `int lpopt::lpoptData::info_iters_since_header_ [private]`

number of iteration summaries actually printed since last summary header was printed

Definition at line 630 of file `lpoptData.hpp`.

6.71.4.27 `SmartPointer<IteratesVectorSpace> lpopt::lpoptData::iterates_space_ [private]`

[VectorSpace](#) for all the iterates.

Definition at line 634 of file `lpoptData.hpp`.

6.71.4.28 `TimingStatistics lpopt::lpoptData::timing_statistics_ [private]`

[TimingStatistics](#) object collecting all [lpopt](#) timing statistics.

Definition at line 638 of file `lpoptData.hpp`.

6.71.4.29 `Number lpopt::lpoptData::cpu_time_start_ [private]`

CPU time counter at initialization.

Definition at line 641 of file `lpoptData.hpp`.

6.71.4.30 `SmartPointer<lpoptAdditionalData> lpopt::lpoptData::add_data_ [private]`

Object for the data specific for the Chen-Goldfarb penalty method algorithm.

Definition at line 645 of file `lpoptData.hpp`.

6.71.4.31 **Number** Ipopt::IpoptData::pd_pert_x_ [private]

Definition at line 650 of file IpoptData.hpp.

6.71.4.32 **Number** Ipopt::IpoptData::pd_pert_s_ [private]

Definition at line 651 of file IpoptData.hpp.

6.71.4.33 **Number** Ipopt::IpoptData::pd_pert_c_ [private]

Definition at line 652 of file IpoptData.hpp.

6.71.4.34 **Number** Ipopt::IpoptData::pd_pert_d_ [private]

Definition at line 653 of file IpoptData.hpp.

The documentation for this class was generated from the following file:

- Algorithm/[IpoptData.hpp](#)

6.72 Ipopt::IpoptException Class Reference

This is the base class for all exceptions.

```
#include <IpException.hpp>
```

Public Member Functions

- void [ReportException](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level=[J_ERROR](#)) const
Method to report the exception to a journalist.
- const std::string & [Message](#) () const

Constructors/Destructors

- [IpoptException](#) (std::string msg, std::string [file_name](#), [Index](#) line_number, std::string type="IpoptException")
Constructor.
- [IpoptException](#) (const [IpoptException](#) ©)
Copy Constructor.
- virtual [~IpoptException](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IpoptException](#) ()
Default Constructor.
- void [operator=](#) (const [IpoptException](#) &)
Overloaded Equals Operator.

Private Attributes

- `std::string msg_`
- `std::string file_name_`
- `Index line_number_`
- `std::string type_`

6.72.1 Detailed Description

This is the base class for all exceptions.

The easiest way to use this class is by means of the following macros:

```
DECLARE_STD_EXCEPTION(ExceptionType);
```

This macro defines a new class with the name `ExceptionType`, inherited from the base class `IpoptException`. After this, exceptions of this type can be thrown using

```
THROW_EXCEPTION(ExceptionType, Message);
```

where `Message` is a `std::string` with a message that gives an indication of what caused the exception. Exceptions can also be thrown using the macro

```
ASSERT_EXCEPTION(Condition, ExceptionType, Message);
```

where `Conditions` is an expression. If `Condition` evaluates to false, then the exception of the type `ExceptionType` is thrown with `Message`.

When an exception is caught, the method `ReportException` can be used to write the information about the exception to the [Journalist](#), using the level `J_ERROR` and the category `J_MAIN`.

Definition at line 57 of file `IpException.hpp`.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 `Ipopt::IpoptException::IpoptException (std::string msg, std::string file_name, Index line_number, std::string type = "IpoptException") [inline]`

Constructor.

Definition at line 63 of file `IpException.hpp`.

6.72.2.2 `Ipopt::IpoptException::IpoptException (const IpoptException & copy) [inline]`

Copy Constructor.

Definition at line 72 of file `IpException.hpp`.

6.72.2.3 `virtual Ipopt::IpoptException::~~IpoptException () [inline],[virtual]`

Default destructor.

Definition at line 81 of file `IpException.hpp`.

6.72.2.4 `Ipopt::IpoptException::IpoptException () [private]`

Default Constructor.

6.72.3 Member Function Documentation

6.72.3.1 `void Ipopt::IpoptException::ReportException (const Journalist & jnlst, EJournalLevel level = J_ERROR) const` `[inline]`

Method to report the exception to a journalist.

Definition at line 86 of file IpException.hpp.

6.72.3.2 `const std::string& Ipopt::IpoptException::Message () const` `[inline]`

Definition at line 94 of file IpException.hpp.

6.72.3.3 `void Ipopt::IpoptException::operator= (const IpoptException &)` `[private]`

Overloaded Equals Operator.

6.72.4 Member Data Documentation

6.72.4.1 `std::string Ipopt::IpoptException::msg_` `[private]`

Definition at line 115 of file IpException.hpp.

6.72.4.2 `std::string Ipopt::IpoptException::file_name_` `[private]`

Definition at line 116 of file IpException.hpp.

6.72.4.3 `Index Ipopt::IpoptException::line_number_` `[private]`

Definition at line 117 of file IpException.hpp.

6.72.4.4 `std::string Ipopt::IpoptException::type_` `[private]`

Definition at line 118 of file IpException.hpp.

The documentation for this class was generated from the following file:

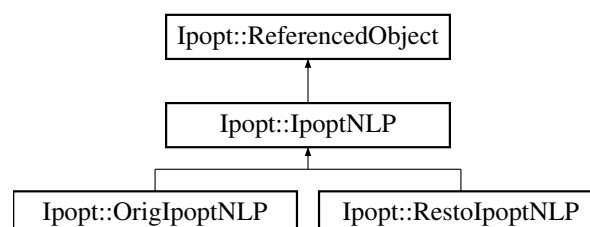
- [Common/IpException.hpp](#)

6.73 Ipopt::IpoptNLP Class Reference

This is the abstract base class for classes that map the traditional [NLP](#) into something that is more useful by [Ipopt](#).

```
#include <IpIpoptNLP.hpp>
```

Inheritance diagram for Ipopt::IpoptNLP:



Public Member Functions

- virtual bool **Initialize** (const **Journalist** &jnlst, const **OptionsList** &options, const std::string &prefix)
Initialization method.
- virtual bool **InitializeStructures** (**SmartPtr**< **Vector** > &x, bool init_x, **SmartPtr**< **Vector** > &y_c, bool init_y_c, **SmartPtr**< **Vector** > &y_d, bool init_y_d, **SmartPtr**< **Vector** > &z_L, bool init_z_L, **SmartPtr**< **Vector** > &z_U, bool init_z_U, **SmartPtr**< **Vector** > &v_L, **SmartPtr**< **Vector** > &v_U)=0
Initialize (create) structures for the iteration data.
- virtual bool **GetWarmStartIterate** (**IteratesVector** &warm_start_iterate)=0
Method accessing the GetWarmStartIterate of the NLP.
- virtual void **GetSpaces** (**SmartPtr**< const **VectorSpace** > &x_space, **SmartPtr**< const **VectorSpace** > &c_space, **SmartPtr**< const **VectorSpace** > &d_space, **SmartPtr**< const **VectorSpace** > &x_l_space, **SmartPtr**< const **MatrixSpace** > &px_l_space, **SmartPtr**< const **VectorSpace** > &x_u_space, **SmartPtr**< const **MatrixSpace** > &px_u_space, **SmartPtr**< const **VectorSpace** > &d_l_space, **SmartPtr**< const **MatrixSpace** > &pd_l_space, **SmartPtr**< const **VectorSpace** > &d_u_space, **SmartPtr**< const **MatrixSpace** > &pd_u_space, **SmartPtr**< const **MatrixSpace** > &Jac_c_space, **SmartPtr**< const **MatrixSpace** > &Jac_d_space, **SmartPtr**< const **SymMatrixSpace** > &Hess_lagrangian_space)=0
Accessor method for vector/matrix spaces pointers.
- virtual void **AdjustVariableBounds** (const **Vector** &new_x_L, const **Vector** &new_x_U, const **Vector** &new_d_L, const **Vector** &new_d_U)=0
Method for adapting the variable bounds.
- **SmartPtr**< **NLPScalingObject** > **NLP_scaling** () const
Returns the scaling strategy object.

Constructors/Destructors

- **IpoptNLP** (const **SmartPtr**< **NLPScalingObject** > nlp_scaling)
- virtual **~IpoptNLP** ()
Default destructor.

Possible Exceptions

- **DECLARE_STD_EXCEPTION** (Eval_Error)
thrown if there is any error evaluating values from the nlp
- virtual **Number** f (const **Vector** &x)=0
Accessor methods for model data.
- virtual **SmartPtr**< const **Vector** > **grad_f** (const **Vector** &x)=0
Gradient of the objective.
- virtual **SmartPtr**< const **Vector** > **c** (const **Vector** &x)=0
Equality constraint residual.
- virtual **SmartPtr**< const **Matrix** > **jac_c** (const **Vector** &x)=0
Jacobian Matrix for equality constraints.
- virtual **SmartPtr**< const **Vector** > **d** (const **Vector** &x)=0
Inequality constraint residual (reformulated as equalities with slacks).
- virtual **SmartPtr**< const **Matrix** > **jac_d** (const **Vector** &x)=0
Jacobian Matrix for inequality constraints.
- virtual **SmartPtr**< const **SymMatrix** > **h** (const **Vector** &x, **Number** obj_factor, const **Vector** &yc, const **Vector** &yd)=0
Hessian of the Lagrangian.

- virtual [SmartPtr](#)< const [Vector](#) > [x_L](#) () const =0
Lower bounds on x.
- virtual [SmartPtr](#)< const [Matrix](#) > [Px_L](#) () const =0
Permutation matrix ($x_L \rightarrow x$)
- virtual [SmartPtr](#)< const [Vector](#) > [x_U](#) () const =0
Upper bounds on x.
- virtual [SmartPtr](#)< const [Matrix](#) > [Px_U](#) () const =0
Permutation matrix ($x_U \rightarrow x$).
- virtual [SmartPtr](#)< const [Vector](#) > [d_L](#) () const =0
Lower bounds on d.
- virtual [SmartPtr](#)< const [Matrix](#) > [Pd_L](#) () const =0
Permutation matrix ($d_L \rightarrow d$)
- virtual [SmartPtr](#)< const [Vector](#) > [d_U](#) () const =0
Upper bounds on d.
- virtual [SmartPtr](#)< const [Matrix](#) > [Pd_U](#) () const =0
Permutation matrix ($d_U \rightarrow d$).
- virtual [SmartPtr](#)< const [SymMatrixSpace](#) > [HessianMatrixSpace](#) () const =0
Accessor method to obtain the [MatrixSpace](#) for the Hessian matrix (or it's approximation)

Counters for the number of function evaluations.

- virtual [Index](#) [f_evals](#) () const =0
- virtual [Index](#) [grad_f_evals](#) () const =0
- virtual [Index](#) [c_evals](#) () const =0
- virtual [Index](#) [jac_c_evals](#) () const =0
- virtual [Index](#) [d_evals](#) () const =0
- virtual [Index](#) [jac_d_evals](#) () const =0
- virtual [Index](#) [h_evals](#) () const =0

Special method for dealing with the fact that the

restoration phase objective function depends on the barrier parameter

- virtual bool [objective_depends_on_mu](#) () const
Method for telling the [IpoptCalculatedQuantities](#) class whether the objective function depends on the barrier function.
- virtual [Number](#) [f](#) (const [Vector](#) &x, [Number](#) mu)=0
Replacement for the default objective function method which knows about the barrier parameter.
- virtual [SmartPtr](#)< const [Vector](#) > [grad_f](#) (const [Vector](#) &x, [Number](#) mu)=0
Replacement for the default objective gradient method which knows about the barrier parameter.
- virtual [SmartPtr](#)< const [SymMatrix](#) > [h](#) (const [Vector](#) &x, [Number](#) obj_factor, const [Vector](#) &yc, const [Vector](#) &yd, [Number](#) mu)=0
Replacement for the default Lagrangian Hessian method which knows about the barrier parameter.
- virtual [SmartPtr](#)< const [SymMatrix](#) > [uninitialized_h](#) ()=0
Provides a Hessian matrix from the correct matrix space with uninitialized values.

solution routines

- virtual void [FinalizeSolution](#) ([SolverReturn](#) status, const [Vector](#) &x, const [Vector](#) &z_L, const [Vector](#) &z_U, const [Vector](#) &c, const [Vector](#) &d, const [Vector](#) &y_c, const [Vector](#) &y_d, [Number](#) obj_value, const [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)=0
- virtual bool [IntermediateCallBack](#) ([AlgorithmMode](#) mode, [Index](#) iter, [Number](#) obj_value, [Number](#) inf_pr, [Number](#) inf_du, [Number](#) mu, [Number](#) d_norm, [Number](#) regularization_size, [Number](#) alpha_du, [Number](#) alpha_pr, [Index](#) ls_trials, [SmartPtr](#)< const [IpoptData](#) > ip_data, [SmartPtr](#)< [IpoptCalculatedQuantities](#) > ip_cq)=0

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `IpoptNLP` (const `IpoptNLP` &)
Copy Constructor.
- void `operator=` (const `IpoptNLP` &)
Overloaded Equals Operator.

Private Attributes

- `SmartPtr< NLPScalingObject > nlp_scaling_`

6.73.1 Detailed Description

This is the abstract base class for classes that map the traditional `NLP` into something that is more useful by `Ipopt`.

This class takes care of storing the calculated model results, handles cacheing, and (some day) takes care of addition of slacks.

Definition at line 28 of file `IpIpoptNLP.hpp`.

6.73.2 Constructor & Destructor Documentation

6.73.2.1 `Ipopt::IpoptNLP::IpoptNLP (const SmartPtr< NLPScalingObject > nlp_scaling) [inline]`

Definition at line 33 of file `IpIpoptNLP.hpp`.

6.73.2.2 `virtual Ipopt::IpoptNLP::~~IpoptNLP () [inline],[virtual]`

Default destructor.

Definition at line 39 of file `IpIpoptNLP.hpp`.

6.73.2.3 `Ipopt::IpoptNLP::IpoptNLP (const IpoptNLP &) [private]`

Copy Constructor.

6.73.3 Member Function Documentation

6.73.3.1 `virtual bool Ipopt::IpoptNLP::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [inline],[virtual]`

Initialization method.

Set the internal options and initialize internal data structures.

Reimplemented in `Ipopt::OrigIpoptNLP`, and `Ipopt::RestIpoptNLP`.

Definition at line 45 of file `IpIpoptNLP.hpp`.

6.73.3.2 Ipopt::IpoptNLP::DECLARE_STD_EXCEPTION (Eval_Error)

thrown if there is any error evaluating values from the nlp

```
6.73.3.3 virtual bool Ipopt::IpoptNLP::InitializeStructures ( SmartPtr< Vector > & x, bool init_x, SmartPtr< Vector > & y_c,
    bool init_y_c, SmartPtr< Vector > & y_d, bool init_y_d, SmartPtr< Vector > & z_L, bool init_z_L, SmartPtr<
    Vector > & z_U, bool init_z_U, SmartPtr< Vector > & v_L, SmartPtr< Vector > & v_U ) [pure virtual]
```

Initialize (create) structures for the iteration data.

Implemented in [Ipopt::OrigIpoptNLP](#), and [Ipopt::RestIpoptNLP](#).

```
6.73.3.4 virtual bool Ipopt::IpoptNLP::GetWarmStartIterate ( IteratesVector & warm_start_iterate ) [pure virtual]
```

Method accessing the GetWarmStartIterate of the [NLP](#).

Implemented in [Ipopt::OrigIpoptNLP](#), and [Ipopt::RestIpoptNLP](#).

```
6.73.3.5 virtual Number Ipopt::IpoptNLP::f ( const Vector & x ) [pure virtual]
```

Accessor methods for model data.

Objective value

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

```
6.73.3.6 virtual SmartPtr<const Vector> Ipopt::IpoptNLP::grad_f ( const Vector & x ) [pure virtual]
```

Gradient of the objective.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

```
6.73.3.7 virtual SmartPtr<const Vector> Ipopt::IpoptNLP::c ( const Vector & x ) [pure virtual]
```

Equality constraint residual.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

```
6.73.3.8 virtual SmartPtr<const Matrix> Ipopt::IpoptNLP::jac_c ( const Vector & x ) [pure virtual]
```

Jacobian [Matrix](#) for equality constraints.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

```
6.73.3.9 virtual SmartPtr<const Vector> Ipopt::IpoptNLP::d ( const Vector & x ) [pure virtual]
```

Inequality constraint residual (reformulated as equalities with slacks).

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

```
6.73.3.10 virtual SmartPtr<const Matrix> Ipopt::IpoptNLP::jac_d ( const Vector & x ) [pure virtual]
```

Jacobian [Matrix](#) for inequality constraints.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

```
6.73.3.11 virtual SmartPtr<const SymMatrix> Ipopt::IpoptNLP::h ( const Vector & x, Number obj_factor, const Vector &
    yc, const Vector & yd ) [pure virtual]
```

Hessian of the Lagrangian.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.12 `virtual SmartPtr<const Vector> lpopt::lpoptNLP::x_L () const [pure virtual]`

Lower bounds on x.

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.13 `virtual SmartPtr<const Matrix> lpopt::lpoptNLP::Px_L () const [pure virtual]`

Permutation matrix ($x_{L_} \rightarrow x$)

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.14 `virtual SmartPtr<const Vector> lpopt::lpoptNLP::x_U () const [pure virtual]`

Upper bounds on x.

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.15 `virtual SmartPtr<const Matrix> lpopt::lpoptNLP::Px_U () const [pure virtual]`

Permutation matrix ($x_{U_} \rightarrow x$).

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.16 `virtual SmartPtr<const Vector> lpopt::lpoptNLP::d_L () const [pure virtual]`

Lower bounds on d.

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.17 `virtual SmartPtr<const Matrix> lpopt::lpoptNLP::Pd_L () const [pure virtual]`

Permutation matrix ($d_{L_} \rightarrow d$)

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.18 `virtual SmartPtr<const Vector> lpopt::lpoptNLP::d_U () const [pure virtual]`

Upper bounds on d.

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.19 `virtual SmartPtr<const Matrix> lpopt::lpoptNLP::Pd_U () const [pure virtual]`

Permutation matrix ($d_{U_} \rightarrow d$).

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.20 `virtual SmartPtr<const SymMatrixSpace> lpopt::lpoptNLP::HessianMatrixSpace () const [pure virtual]`

Accessor method to obtain the [MatrixSpace](#) for the Hessian matrix (or it's approximation)

Implemented in [lpopt::RestolpoptNLP](#), and [lpopt::OriglpoptNLP](#).

6.73.3.21 `virtual void Ipopt::IpoptNLP::GetSpaces (SmartPtr< const VectorSpace > & x_space, SmartPtr< const VectorSpace > & c_space, SmartPtr< const VectorSpace > & d_space, SmartPtr< const VectorSpace > & x_l_space, SmartPtr< const MatrixSpace > & px_l_space, SmartPtr< const VectorSpace > & x_u_space, SmartPtr< const MatrixSpace > & px_u_space, SmartPtr< const VectorSpace > & d_l_space, SmartPtr< const MatrixSpace > & pd_l_space, SmartPtr< const VectorSpace > & d_u_space, SmartPtr< const MatrixSpace > & pd_u_space, SmartPtr< const MatrixSpace > & Jac_c_space, SmartPtr< const MatrixSpace > & Jac_d_space, SmartPtr< const SymMatrixSpace > & Hess_lagrangian_space) [pure virtual]`

Accessor method for vector/matrix spaces pointers.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.22 `virtual void Ipopt::IpoptNLP::AdjustVariableBounds (const Vector & new_x_L, const Vector & new_x_U, const Vector & new_d_L, const Vector & new_d_U) [pure virtual]`

Method for adapting the variable bounds.

This is called if slacks are becoming too small

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.23 `virtual Index Ipopt::IpoptNLP::f_evals () const [pure virtual]`

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.24 `virtual Index Ipopt::IpoptNLP::grad_f_evals () const [pure virtual]`

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.25 `virtual Index Ipopt::IpoptNLP::c_evals () const [pure virtual]`

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.26 `virtual Index Ipopt::IpoptNLP::jac_c_evals () const [pure virtual]`

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.27 `virtual Index Ipopt::IpoptNLP::d_evals () const [pure virtual]`

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.28 `virtual Index Ipopt::IpoptNLP::jac_d_evals () const [pure virtual]`

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.29 `virtual Index Ipopt::IpoptNLP::h_evals () const [pure virtual]`

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.30 `virtual bool Ipopt::IpoptNLP::objective_depends_on_mu () const [inline],[virtual]`

Method for telling the [IpoptCalculatedQuantities](#) class whether the objective function depends on the barrier function.

This is only used for the restoration phase NLP formulation. Probably only [RestIpoptNLP](#) should overwrite this.

Reimplemented in [Ipopt::RestIpoptNLP](#).

Definition at line 180 of file `IpoptNLP.hpp`.

6.73.3.31 `virtual Number Ipopt::IpoptNLP::f (const Vector & x, Number mu) [pure virtual]`

Replacement for the default objective function method which knows about the barrier parameter.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.32 `virtual SmartPtr<const Vector> Ipopt::IpoptNLP::grad_f (const Vector & x, Number mu) [pure virtual]`

Replacement for the default objective gradient method which knows about the barrier parameter.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.33 `virtual SmartPtr<const SymMatrix> Ipopt::IpoptNLP::h (const Vector & x, Number obj_factor, const Vector & yc, const Vector & yd, Number mu) [pure virtual]`

Replacement for the default Lagrangian Hessian method which knows about the barrier parameter.

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.34 `virtual SmartPtr<const SymMatrix> Ipopt::IpoptNLP::uninitialized_h () [pure virtual]`

Provides a Hessian matrix from the correct matrix space with uninitialized values.

This can be used in LeastSquareMults to obtain a "zero Hessian".

Implemented in [Ipopt::RestIpoptNLP](#), and [Ipopt::OrigIpoptNLP](#).

6.73.3.35 `virtual void Ipopt::IpoptNLP::FinalizeSolution (SolverReturn status, const Vector & x, const Vector & z_L, const Vector & z_U, const Vector & c, const Vector & d, const Vector & y_c, const Vector & y_d, Number obj_value, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [pure virtual]`

Implemented in [Ipopt::OrigIpoptNLP](#), and [Ipopt::RestIpoptNLP](#).

6.73.3.36 `virtual bool Ipopt::IpoptNLP::IntermediateCallBack (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, SmartPtr<const IpoptData> ip_data, SmartPtr<IpoptCalculatedQuantities> ip_cq) [pure virtual]`

Implemented in [Ipopt::OrigIpoptNLP](#), and [Ipopt::RestIpoptNLP](#).

6.73.3.37 `SmartPtr<NLPScalingObject> Ipopt::IpoptNLP::NLP_scaling () const [inline]`

Returns the scaling strategy object.

Definition at line 229 of file [IpoptNLP.hpp](#).

6.73.3.38 `void Ipopt::IpoptNLP::operator=(const IpoptNLP &) [private]`

Overloaded Equals Operator.

6.73.4 Member Data Documentation

6.73.4.1 `SmartPtr<NLPScalingObject> Ipopt::IpoptNLP::nlp_scaling_ [private]`

Definition at line 253 of file [IpoptNLP.hpp](#).

The documentation for this class was generated from the following file:

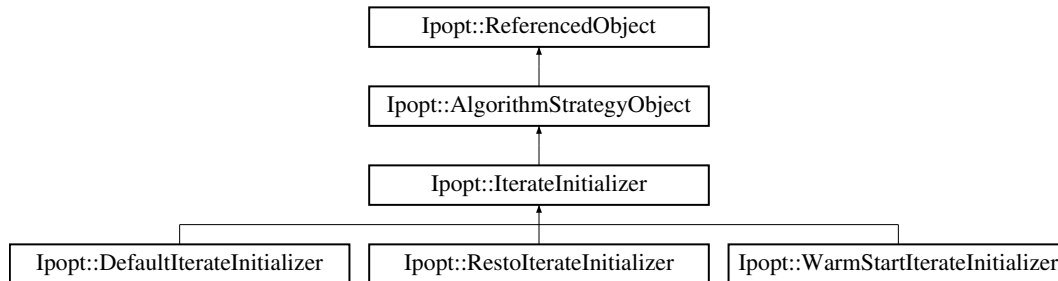
- Algorithm/[IpIpoptNLP.hpp](#)

6.74 Ipopt::IterateInitializer Class Reference

Base class for all methods for initializing the iterates.

```
#include <IpIterateInitializer.hpp>
```

Inheritance diagram for Ipopt::IterateInitializer:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [SetInitialIterates](#) ()=0
Compute the initial iterates and set the into the curr field of the ip_data object.

Constructors/Destructors

- [IterateInitializer](#) ()
Default Constructor.
- virtual [~IterateInitializer](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IterateInitializer](#) (const [IterateInitializer](#) &)
Copy Constructor.
- void [operator=](#) (const [IterateInitializer](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.74.1 Detailed Description

Base class for all methods for initializing the iterates.

Definition at line 22 of file `IpIterateInitializer.hpp`.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 `Ipopt::IterateInitializer::IterateInitializer ()` `[inline]`

Default Constructor.

Definition at line 28 of file `IpIterateInitializer.hpp`.

6.74.2.2 `virtual Ipopt::IterateInitializer::~~IterateInitializer ()` `[inline]`, `[virtual]`

Default destructor.

Definition at line 32 of file `IpIterateInitializer.hpp`.

6.74.2.3 `Ipopt::IterateInitializer::IterateInitializer (const IterateInitializer &)` `[private]`

Copy Constructor.

6.74.3 Member Function Documentation

6.74.3.1 `virtual bool Ipopt::IterateInitializer::InitializeImpl (const OptionsList & options, const std::string & prefix)` `[pure virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::DefaultIterateInitializer](#), [Ipopt::RestOfIterateInitializer](#), and [Ipopt::WarmStartIterateInitializer](#).

6.74.3.2 `virtual bool Ipopt::IterateInitializer::SetInitialIterates ()` `[pure virtual]`

Compute the initial iterates and set the into the `curr` field of the `ip_data` object.

Implemented in [Ipopt::DefaultIterateInitializer](#), [Ipopt::RestOfIterateInitializer](#), and [Ipopt::WarmStartIterateInitializer](#).

6.74.3.3 `void Ipopt::IterateInitializer::operator= (const IterateInitializer &)` `[private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

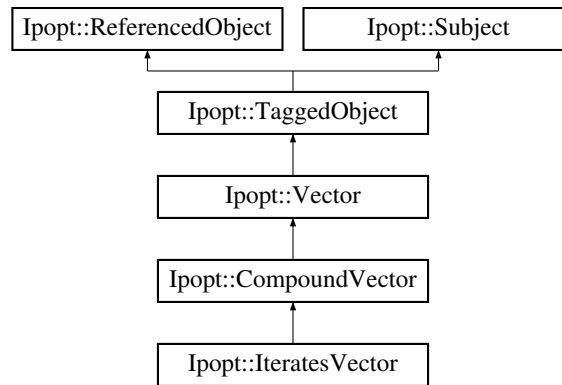
- [Algorithm/IpIterateInitializer.hpp](#)

6.75 Ipopt::IteratesVector Class Reference

Specialized [CompoundVector](#) class specifically for the algorithm iterates.

```
#include <IpIteratesVector.hpp>
```

Inheritance diagram for `Ipopt::IteratesVector`:



Public Member Functions

- [IteratesVector](#) (const [IteratesVectorSpace](#) *owner_space, bool create_new)
Constructors / Destructors.
- virtual [~IteratesVector](#) ()
- [SmartPointer< IteratesVector > MakeNewIteratesVector](#) (bool create_new=true) const
Make New methods.
- [SmartPointer< IteratesVector > MakeNewIteratesVectorCopy](#) () const
Use this method to create a new iterates vector with a copy of all the data.
- [SmartPointer< IteratesVector > MakeNewContainer](#) () const
Use this method to create a new iterates vector container.
- [SmartPointer< const Vector > x](#) () const
Iterates Set/Get Methods.
- [SmartPointer< Vector > x_NonConst](#) ()
Get the x iterate (non-const) - this can only be called if the vector was created intenally, or the Set_x_NonConst method was used.
- [SmartPointer< Vector > create_new_x](#) ()
Create a new vector in the x entry.
- [SmartPointer< Vector > create_new_x_copy](#) ()
Create a new vector in the x entry and copy the current values into it.
- void [Set_x](#) (const [Vector](#) &vec)
Set the x iterate (const).
- void [Set_x_NonConst](#) ([Vector](#) &vec)
Set the x iterate (non-const).
- [SmartPointer< const Vector > s](#) () const
Get the s iterate (const)
- [SmartPointer< Vector > s_NonConst](#) ()
Get the s iterate (non-const) - this can only be called if the vector was created intenally, or the Set_s_NonConst method was used.
- [SmartPointer< Vector > create_new_s](#) ()
Create a new vector in the s entry.
- [SmartPointer< Vector > create_new_s_copy](#) ()
Create a new vector in the s entry and copy the current values into it.

- void `Set_s` (const `Vector` &vec)
Set the s iterate (const).
- void `Set_s_NonConst` (`Vector` &vec)
Set the s iterate (non-const).
- `SmartPtr`< const `Vector` > `y_c` () const
Get the y_c iterate (const)
- `SmartPtr`< `Vector` > `y_c_NonConst` ()
Get the y_c iterate (non-const) - this can only be called if the vector was created internally, or the Set_y_c_NonConst method was used.
- `SmartPtr`< `Vector` > `create_new_y_c` ()
Create a new vector in the y_c entry.
- `SmartPtr`< `Vector` > `create_new_y_c_copy` ()
Create a new vector in the y_c entry and copy the current values into it.
- void `Set_y_c` (const `Vector` &vec)
Set the y_c iterate (const).
- void `Set_y_c_NonConst` (`Vector` &vec)
Set the y_c iterate (non-const).
- `SmartPtr`< const `Vector` > `y_d` () const
Get the y_d iterate (const)
- `SmartPtr`< `Vector` > `y_d_NonConst` ()
Get the y_d iterate (non-const) - this can only be called if the vector was created internally, or the Set_y_d_NonConst method was used.
- `SmartPtr`< `Vector` > `create_new_y_d` ()
Create a new vector in the y_d entry.
- `SmartPtr`< `Vector` > `create_new_y_d_copy` ()
Create a new vector in the y_d entry and copy the current values into it.
- void `Set_y_d` (const `Vector` &vec)
Set the y_d iterate (const).
- void `Set_y_d_NonConst` (`Vector` &vec)
Set the y_d iterate (non-const).
- `SmartPtr`< const `Vector` > `z_L` () const
Get the z_L iterate (const)
- `SmartPtr`< `Vector` > `z_L_NonConst` ()
Get the z_L iterate (non-const) - this can only be called if the vector was created internally, or the Set_z_L_NonConst method was used.
- `SmartPtr`< `Vector` > `create_new_z_L` ()
Create a new vector in the z_L entry.
- `SmartPtr`< `Vector` > `create_new_z_L_copy` ()
Create a new vector in the z_L entry and copy the current values into it.
- void `Set_z_L` (const `Vector` &vec)
Set the z_L iterate (const).
- void `Set_z_L_NonConst` (`Vector` &vec)
Set the z_L iterate (non-const).
- `SmartPtr`< const `Vector` > `z_U` () const
Get the z_U iterate (const)
- `SmartPtr`< `Vector` > `z_U_NonConst` ()
Get the z_U iterate (non-const) - this can only be called if the vector was created internally, or the Set_z_U_NonConst method was used.

- `SmartPtr< Vector > create_new_z_U ()`
Create a new vector in the `z_U` entry.
- `SmartPtr< Vector > create_new_z_U_copy ()`
Create a new vector in the `z_U` entry and copy the current values into it.
- `void Set_z_U (const Vector &vec)`
Set the `z_U` iterate (const).
- `void Set_z_U_NonConst (Vector &vec)`
Set the `z_U` iterate (non-const).
- `SmartPtr< const Vector > v_L () const`
Get the `v_L` iterate (const)
- `SmartPtr< Vector > v_L_NonConst ()`
Get the `v_L` iterate (non-const) - this can only be called if the vector was created internally, or the `Set_v_L_NonConst` method was used.
- `SmartPtr< Vector > create_new_v_L ()`
Create a new vector in the `v_L` entry.
- `SmartPtr< Vector > create_new_v_L_copy ()`
Create a new vector in the `v_L` entry and copy the current values into it.
- `void Set_v_L (const Vector &vec)`
Set the `v_L` iterate (const).
- `void Set_v_L_NonConst (Vector &vec)`
Set the `v_L` iterate (non-const).
- `SmartPtr< const Vector > v_U () const`
Get the `v_U` iterate (const)
- `SmartPtr< Vector > v_U_NonConst ()`
Get the `v_U` iterate (non-const) - this can only be called if the vector was created internally, or the `Set_v_U_NonConst` method was used.
- `SmartPtr< Vector > create_new_v_U ()`
Create a new vector in the `v_U` entry.
- `SmartPtr< Vector > create_new_v_U_copy ()`
Create a new vector in the `v_U` entry and copy the current values into it.
- `void Set_v_U (const Vector &vec)`
Set the `v_U` iterate (const).
- `void Set_v_U_NonConst (Vector &vec)`
Set the `v_U` iterate (non-const).
- `void Set_primal (const Vector &x, const Vector &s)`
Set the primal variables all in one shot.
- `void Set_primal_NonConst (Vector &x, Vector &s)`
- `void Set_eq_mult (const Vector &y_c, const Vector &y_d)`
Set the eq multipliers all in one shot.
- `void Set_eq_mult_NonConst (Vector &y_c, Vector &y_d)`
- `void Set_bound_mult (const Vector &z_L, const Vector &z_U, const Vector &v_L, const Vector &v_U)`
Set the bound multipliers all in one shot.
- `void Set_bound_mult_NonConst (Vector &z_L, Vector &z_U, Vector &v_L, Vector &v_U)`
- `TaggedObject::Tag GetTagSum () const`
Get a sum of the tags of the contained items.

Private Member Functions

- `SmartPtr< const Vector > GetIterateFromComp (Index i) const`
private method to return the const element from the compound vector.
- `SmartPtr< Vector > GetNonConstIterateFromComp (Index i)`
private method to return the non-const element from the compound vector.

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `IteratesVector ()`
Default Constructor.
- `IteratesVector (const IteratesVector &)`
Copy Constructor.
- `void operator= (const IteratesVector &)`
Overloaded Equals Operator.

Private Attributes

- `const IteratesVectorSpace * owner_space_`

Additional Inherited Members

6.75.1 Detailed Description

Specialized `CompoundVector` class specifically for the algorithm iterates.

This class inherits from `CompoundVector` and is a specialized class for handling the iterates of the `Ipopt` Algorithm, that is, `x`, `s`, `y_c`, `y_d`, `z_L`, `z_U`, `v_L`, and `v_U`. It inherits from `CompoundVector` so it can behave like a CV in most calculations, but it has fixed dimensions and cannot be customized

Definition at line 27 of file `IpIteratesVector.hpp`.

6.75.2 Constructor & Destructor Documentation

6.75.2.1 `Ipopt::IteratesVector::IteratesVector (const IteratesVectorSpace * owner_space, bool create_new)`

Constructors / Destructors.

6.75.2.2 `virtual Ipopt::IteratesVector::~~IteratesVector () [virtual]`

6.75.2.3 `Ipopt::IteratesVector::IteratesVector () [private]`

Default Constructor.

6.75.2.4 `Ipopt::IteratesVector::IteratesVector (const IteratesVector &) [private]`

Copy Constructor.

6.75.3 Member Function Documentation

6.75.3.1 **SmartPointer<IteratesVector> Ipopt::IteratesVector::MakeNewIteratesVector (bool *create_new* = true) const**

Make New methods.

Use this method to create a new iterates vector. The MakeNew method on the [Vector](#) class also works, but it does not give the *create_new* option.

6.75.3.2 **SmartPointer<IteratesVector> Ipopt::IteratesVector::MakeNewIteratesVectorCopy () const** `[inline]`

Use this method to create a new iterates vector with a copy of all the data.

Definition at line 48 of file `IpIteratesVector.hpp`.

6.75.3.3 **SmartPointer<IteratesVector> Ipopt::IteratesVector::MakeNewContainer () const**

Use this method to create a new iterates vector container.

This creates a new NonConst container, but the elements inside the iterates vector may be const. Therefore, the container can be modified to point to new entries, but the existing entries may or may not be modifiable.

6.75.3.4 **SmartPointer<const Vector> Ipopt::IteratesVector::x () const** `[inline]`

Iterates Set/Get Methods.

Get the x iterate (const)

Definition at line 67 of file `IpIteratesVector.hpp`.

6.75.3.5 **SmartPointer<Vector> Ipopt::IteratesVector::x_NonConst ()** `[inline]`

Get the x iterate (non-const) - this can only be called if the vector was created internally, or the `Set_x_NonConst` method was used.

Definition at line 75 of file `IpIteratesVector.hpp`.

6.75.3.6 **SmartPointer<Vector> Ipopt::IteratesVector::create_new_x ()** `[inline]`

Create a new vector in the x entry.

Definition at line 639 of file `IpIteratesVector.hpp`.

6.75.3.7 **SmartPointer<Vector> Ipopt::IteratesVector::create_new_x_copy ()** `[inline]`

Create a new vector in the x entry and copy the current values into it.

Definition at line 86 of file `IpIteratesVector.hpp`.

6.75.3.8 **void Ipopt::IteratesVector::Set_x (const Vector & vec)** `[inline]`

Set the x iterate (const).

Sets the pointer, does NOT copy data.

Definition at line 96 of file `IpIteratesVector.hpp`.

6.75.3.9 **void Ipopt::IteratesVector::Set_x_NonConst (Vector & vec)** `[inline]`

Set the x iterate (non-const).

Sets the pointer, does NOT copy data.

Definition at line 103 of file `IpIteratesVector.hpp`.

6.75.3.10 `SmartPtr<const Vector> Ipopt::IteratesVector::s () const` `[inline]`

Get the `s` iterate (const)

Definition at line 109 of file `IpIteratesVector.hpp`.

6.75.3.11 `SmartPtr<Vector> Ipopt::IteratesVector::s_NonConst ()` `[inline]`

Get the `s` iterate (non-const) - this can only be called if the vector was created internally, or the `Set_s_NonConst` method was used.

Definition at line 117 of file `IpIteratesVector.hpp`.

6.75.3.12 `SmartPtr< Vector > Ipopt::IteratesVector::create_new_s ()` `[inline]`

Create a new vector in the `s` entry.

Definition at line 645 of file `IpIteratesVector.hpp`.

6.75.3.13 `SmartPtr<Vector> Ipopt::IteratesVector::create_new_s_copy ()` `[inline]`

Create a new vector in the `s` entry and copy the current values into it.

Definition at line 128 of file `IpIteratesVector.hpp`.

6.75.3.14 `void Ipopt::IteratesVector::Set_s(const Vector & vec)` `[inline]`

Set the `s` iterate (const).

Sets the pointer, does NOT copy data.

Definition at line 138 of file `IpIteratesVector.hpp`.

6.75.3.15 `void Ipopt::IteratesVector::Set_s_NonConst(Vector & vec)` `[inline]`

Set the `s` iterate (non-const).

Sets the pointer, does NOT copy data.

Definition at line 145 of file `IpIteratesVector.hpp`.

6.75.3.16 `SmartPtr<const Vector> Ipopt::IteratesVector::y_c () const` `[inline]`

Get the `y_c` iterate (const)

Definition at line 151 of file `IpIteratesVector.hpp`.

6.75.3.17 `SmartPtr<Vector> Ipopt::IteratesVector::y_c_NonConst ()` `[inline]`

Get the `y_c` iterate (non-const) - this can only be called if the vector was created internally, or the `Set_y_c_NonConst` method was used.

Definition at line 159 of file `IpIteratesVector.hpp`.

6.75.3.18 `SmartPtr< Vector > Ipopt::IteratesVector::create_new_y_c ()` `[inline]`

Create a new vector in the `y_c` entry.

Definition at line 651 of file `IpIteratesVector.hpp`.

6.75.3.19 SmartPtr<Vector> Ipopt::IteratesVector::create_new_y_c_copy () [inline]

Create a new vector in the y_c entry and copy the current values into it.

Definition at line 170 of file IpIteratesVector.hpp.

6.75.3.20 void Ipopt::IteratesVector::Set_y_c (const Vector & vec) [inline]

Set the y_c iterate (const).

Sets the pointer, does NOT copy data.

Definition at line 180 of file IpIteratesVector.hpp.

6.75.3.21 void Ipopt::IteratesVector::Set_y_c_NonConst (Vector & vec) [inline]

Set the y_c iterate (non-const).

Sets the pointer, does NOT copy data.

Definition at line 187 of file IpIteratesVector.hpp.

6.75.3.22 SmartPtr<const Vector> Ipopt::IteratesVector::y_d () const [inline]

Get the y_d iterate (const)

Definition at line 193 of file IpIteratesVector.hpp.

6.75.3.23 SmartPtr<Vector> Ipopt::IteratesVector::y_d_NonConst () [inline]

Get the y_d iterate (non-const) - this can only be called if the vector was created internally, or the Set_y_d_NonConst method was used.

Definition at line 201 of file IpIteratesVector.hpp.

6.75.3.24 SmartPtr< Vector > Ipopt::IteratesVector::create_new_y_d () [inline]

Create a new vector in the y_d entry.

Definition at line 657 of file IpIteratesVector.hpp.

6.75.3.25 SmartPtr<Vector> Ipopt::IteratesVector::create_new_y_d_copy () [inline]

Create a new vector in the y_d entry and copy the current values into it.

Definition at line 212 of file IpIteratesVector.hpp.

6.75.3.26 void Ipopt::IteratesVector::Set_y_d (const Vector & vec) [inline]

Set the y_d iterate (const).

Sets the pointer, does NOT copy data.

Definition at line 222 of file IpIteratesVector.hpp.

6.75.3.27 void Ipopt::IteratesVector::Set_y_d_NonConst (Vector & vec) [inline]

Set the y_d iterate (non-const).

Sets the pointer, does NOT copy data.

Definition at line 229 of file IpIteratesVector.hpp.

6.75.3.28 `SmartPointer<const Vector> lpopt::IteratesVector::z_L () const` `[inline]`

Get the z_L iterate (const)

Definition at line 235 of file `lpIteratesVector.hpp`.

6.75.3.29 `SmartPointer<Vector> lpopt::IteratesVector::z_L_NonConst ()` `[inline]`

Get the z_L iterate (non-const) - this can only be called if the vector was created internally, or the `Set_z_L_NonConst` method was used.

Definition at line 243 of file `lpIteratesVector.hpp`.

6.75.3.30 `SmartPointer< Vector > lpopt::IteratesVector::create_new_z_L ()` `[inline]`

Create a new vector in the z_L entry.

Definition at line 663 of file `lpIteratesVector.hpp`.

6.75.3.31 `SmartPointer<Vector> lpopt::IteratesVector::create_new_z_L_copy ()` `[inline]`

Create a new vector in the z_L entry and copy the current values into it.

Definition at line 254 of file `lpIteratesVector.hpp`.

6.75.3.32 `void lpopt::IteratesVector::Set_z_L (const Vector & vec)` `[inline]`

Set the z_L iterate (const).

Sets the pointer, does NOT copy data.

Definition at line 264 of file `lpIteratesVector.hpp`.

6.75.3.33 `void lpopt::IteratesVector::Set_z_L_NonConst (Vector & vec)` `[inline]`

Set the z_L iterate (non-const).

Sets the pointer, does NOT copy data.

Definition at line 271 of file `lpIteratesVector.hpp`.

6.75.3.34 `SmartPointer<const Vector> lpopt::IteratesVector::z_U () const` `[inline]`

Get the z_U iterate (const)

Definition at line 277 of file `lpIteratesVector.hpp`.

6.75.3.35 `SmartPointer<Vector> lpopt::IteratesVector::z_U_NonConst ()` `[inline]`

Get the z_U iterate (non-const) - this can only be called if the vector was created internally, or the `Set_z_U_NonConst` method was used.

Definition at line 285 of file `lpIteratesVector.hpp`.

6.75.3.36 `SmartPointer< Vector > lpopt::IteratesVector::create_new_z_U ()` `[inline]`

Create a new vector in the z_U entry.

Definition at line 669 of file `lpIteratesVector.hpp`.

6.75.3.37 SmartPtr<Vector> Ipopt::IteratesVector::create_new_z_U_copy () [inline]

Create a new vector in the z_U entry and copy the current values into it.

Definition at line 296 of file IpIteratesVector.hpp.

6.75.3.38 void Ipopt::IteratesVector::Set_z_U (const Vector & vec) [inline]

Set the z_U iterate (const).

Sets the pointer, does NOT copy data.

Definition at line 306 of file IpIteratesVector.hpp.

6.75.3.39 void Ipopt::IteratesVector::Set_z_U_NonConst (Vector & vec) [inline]

Set the z_U iterate (non-const).

Sets the pointer, does NOT copy data.

Definition at line 313 of file IpIteratesVector.hpp.

6.75.3.40 SmartPtr<const Vector> Ipopt::IteratesVector::v_L () const [inline]

Get the v_L iterate (const)

Definition at line 319 of file IpIteratesVector.hpp.

6.75.3.41 SmartPtr<Vector> Ipopt::IteratesVector::v_L_NonConst () [inline]

Get the v_L iterate (non-const) - this can only be called if the vector was created internally, or the Set_v_L_NonConst method was used.

Definition at line 327 of file IpIteratesVector.hpp.

6.75.3.42 SmartPtr< Vector > Ipopt::IteratesVector::create_new_v_L () [inline]

Create a new vector in the v_L entry.

Definition at line 675 of file IpIteratesVector.hpp.

6.75.3.43 SmartPtr<Vector> Ipopt::IteratesVector::create_new_v_L_copy () [inline]

Create a new vector in the v_L entry and copy the current values into it.

Definition at line 338 of file IpIteratesVector.hpp.

6.75.3.44 void Ipopt::IteratesVector::Set_v_L (const Vector & vec) [inline]

Set the v_L iterate (const).

Sets the pointer, does NOT copy data.

Definition at line 348 of file IpIteratesVector.hpp.

6.75.3.45 void Ipopt::IteratesVector::Set_v_L_NonConst (Vector & vec) [inline]

Set the v_L iterate (non-const).

Sets the pointer, does NOT copy data.

Definition at line 355 of file IpIteratesVector.hpp.

6.75.3.46 `SmartPointer<const Vector> lpopt::IteratesVector::v_U () const` [inline]

Get the v_U iterate (const)

Definition at line 361 of file `lpIteratesVector.hpp`.

6.75.3.47 `SmartPointer<Vector> lpopt::IteratesVector::v_U_NonConst ()` [inline]

Get the v_U iterate (non-const) - this can only be called if the vector was created internally, or the `Set_v_U_NonConst` method was used.

Definition at line 369 of file `lpIteratesVector.hpp`.

6.75.3.48 `SmartPointer< Vector > lpopt::IteratesVector::create_new_v_U ()` [inline]

Create a new vector in the v_U entry.

Definition at line 681 of file `lpIteratesVector.hpp`.

6.75.3.49 `SmartPointer<Vector> lpopt::IteratesVector::create_new_v_U_copy ()` [inline]

Create a new vector in the v_U entry and copy the current values into it.

Definition at line 380 of file `lpIteratesVector.hpp`.

6.75.3.50 `void lpopt::IteratesVector::Set_v_U (const Vector & vec)` [inline]

Set the v_U iterate (const).

Sets the pointer, does NOT copy data.

Definition at line 390 of file `lpIteratesVector.hpp`.

6.75.3.51 `void lpopt::IteratesVector::Set_v_U_NonConst (Vector & vec)` [inline]

Set the v_U iterate (non-const).

Sets the pointer, does NOT copy data.

Definition at line 397 of file `lpIteratesVector.hpp`.

6.75.3.52 `void lpopt::IteratesVector::Set_primal (const Vector & x, const Vector & s)` [inline]

Set the primal variables all in one shot.

Sets the pointers, does NOT copy data

Definition at line 404 of file `lpIteratesVector.hpp`.

6.75.3.53 `void lpopt::IteratesVector::Set_primal_NonConst (Vector & x, Vector & s)` [inline]

Definition at line 409 of file `lpIteratesVector.hpp`.

6.75.3.54 `void lpopt::IteratesVector::Set_eq_mult (const Vector & y_c, const Vector & y_d)` [inline]

Set the eq multipliers all in one shot.

Sets the pointers, does not copy data.

Definition at line 417 of file `lpIteratesVector.hpp`.

6.75.3.55 `void Ipopt::IteratesVector::Set_eq_mult_NonConst (Vector & y_c, Vector & y_d) [inline]`

Definition at line 422 of file `IpIteratesVector.hpp`.

6.75.3.56 `void Ipopt::IteratesVector::Set_bound_mult (const Vector & z_L, const Vector & z_U, const Vector & v_L, const Vector & v_U) [inline]`

Set the bound multipliers all in one shot.

Sets the pointers, does not copy data.

Definition at line 430 of file `IpIteratesVector.hpp`.

6.75.3.57 `void Ipopt::IteratesVector::Set_bound_mult_NonConst (Vector & z_L, Vector & z_U, Vector & v_L, Vector & v_U) [inline]`

Definition at line 437 of file `IpIteratesVector.hpp`.

6.75.3.58 `TaggedObject::Tag Ipopt::IteratesVector::GetTagSum () const [inline]`

Get a sum of the tags of the contained items.

There is no guarantee that this is unique, but there is a high chance it is unique and it can be used for debug checks relatively reliably.

Definition at line 450 of file `IpIteratesVector.hpp`.

6.75.3.59 `void Ipopt::IteratesVector::operator= (const IteratesVector &) [private]`

Overloaded Equals Operator.

6.75.3.60 `SmartPtr<const Vector> Ipopt::IteratesVector::GetIterateFromComp (Index i) const [inline], [private]`

private method to return the const element from the compound vector.

This method will return NULL if none is currently set.

Definition at line 506 of file `IpIteratesVector.hpp`.

6.75.3.61 `SmartPtr<Vector> Ipopt::IteratesVector::GetNonConstIterateFromComp (Index i) [inline], [private]`

private method to return the non-const element from the compound vector.

This method will return NULL if none is currently set.

Definition at line 518 of file `IpIteratesVector.hpp`.

6.75.4 Member Data Documentation

6.75.4.1 `const IteratesVectorSpace* Ipopt::IteratesVector::owner_space_ [private]`

Definition at line 500 of file `IpIteratesVector.hpp`.

The documentation for this class was generated from the following file:

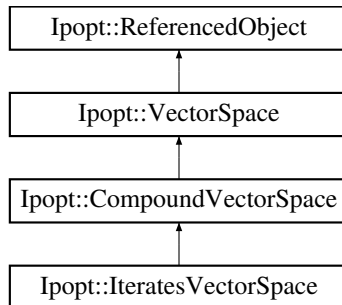
- [Algorithm/IpIteratesVector.hpp](#)

6.76 Ipopt::IteratesVectorSpace Class Reference

Vector Space for the [IteratesVector](#) class.

```
#include <IpIteratesVector.hpp>
```

Inheritance diagram for Ipopt::IteratesVectorSpace:



Public Member Functions

- virtual void [SetCompSpace](#) ([Index](#) icomp, const [VectorSpace](#) &vec_space)

This method hides the [CompoundVectorSpace::SetCompSpace](#) method since the components of the Iterates are fixed at construction.

Constructors/Destructors.

- [IteratesVectorSpace](#) (const [VectorSpace](#) &x_space, const [VectorSpace](#) &s_space, const [VectorSpace](#) &y_c_space, const [VectorSpace](#) &y_d_space, const [VectorSpace](#) &z_L_space, const [VectorSpace](#) &z_U_space, const [VectorSpace](#) &v_L_space, const [VectorSpace](#) &v_U_space)

Constructor that takes the spaces for each of the iterates.

- virtual [~IteratesVectorSpace](#) ()
- virtual [IteratesVector](#) * [MakeNewIteratesVector](#) (bool create_new=true) const
Method for creating vectors .
- const [SmartPtr](#)< const [IteratesVector](#) > [MakeNewIteratesVector](#) (const [Vector](#) &x, const [Vector](#) &s, const [Vector](#) &y_c, const [Vector](#) &y_d, const [Vector](#) &z_L, const [Vector](#) &z_U, const [Vector](#) &v_L, const [Vector](#) &v_U)
Use this method to create a new const [IteratesVector](#).
- virtual [CompoundVector](#) * [MakeNewCompoundVector](#) (bool create_new=true) const
This method overloads [CompoundVectorSpace::MakeNewCompoundVector](#) to make sure that we get a vector of the correct type.
- virtual [Vector](#) * [MakeNew](#) () const
This method creates a new vector (and allocates space in all the contained vectors).

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IteratesVectorSpace](#) ()
Default constructor.
- [IteratesVectorSpace](#) (const [IteratesVectorSpace](#) &)
Copy Constructor.
- [IteratesVectorSpace](#) & [operator=](#) (const [IteratesVectorSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< const [VectorSpace](#) > [x_space_](#)
Contained Spaces.
- [SmartPtr](#)< const [VectorSpace](#) > [s_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [y_c_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [y_d_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [z_L_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [z_U_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [v_L_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [v_U_space_](#)

6.76.1 Detailed Description

[Vector](#) Space for the [IteratesVector](#) class.

This is a specialized vector space for the [IteratesVector](#) class.

Definition at line 531 of file `IpIteratesVector.hpp`.

6.76.2 Constructor & Destructor Documentation

6.76.2.1 `Ipopt::IteratesVectorSpace::IteratesVectorSpace (const VectorSpace & x_space, const VectorSpace & s_space, const VectorSpace & y_c_space, const VectorSpace & y_d_space, const VectorSpace & z_L_space, const VectorSpace & z_U_space, const VectorSpace & v_L_space, const VectorSpace & v_U_space)`

Constructor that takes the spaces for each of the iterates.

Warning! None of these can be NULL !

6.76.2.2 `virtual Ipopt::IteratesVectorSpace::~~IteratesVectorSpace () [virtual]`

6.76.2.3 `Ipopt::IteratesVectorSpace::IteratesVectorSpace () [private]`

Default constructor.

6.76.2.4 `Ipopt::IteratesVectorSpace::IteratesVectorSpace (const IteratesVectorSpace &) [private]`

Copy Constructor.

6.76.3 Member Function Documentation

6.76.3.1 `virtual IteratesVector* Ipopt::IteratesVectorSpace::MakeNewIteratesVector (bool create_new = true) const [inline], [virtual]`

Method for creating vectors .

Use this to create a new [IteratesVector](#). You can pass-in `create_new = false` if you only want a container and do not want vectors allocated.

Definition at line 554 of file `IpIteratesVector.hpp`.

```
6.76.3.2  const SmartPtr<const IteratesVector> Ipopt::IteratesVectorSpace::MakeNewIteratesVector ( const Vector & x, const
        Vector & s, const Vector & y_c, const Vector & y_d, const Vector & z_L, const Vector & z_U, const Vector & v_L,
        const Vector & v_U ) [inline]
```

Use this method to create a new `const` [IteratesVector](#).

You must pass in valid pointers for all of the entries.

Definition at line 562 of file `IpIteratesVector.hpp`.

```
6.76.3.3  virtual CompoundVector* Ipopt::IteratesVectorSpace::MakeNewCompoundVector ( bool create_new = true ) const
        [inline], [virtual]
```

This method overloads `CompoundVectorSpace::MakeNewCompoundVector` to make sure that we get a vector of the correct type.

Reimplemented from [Ipopt::CompoundVectorSpace](#).

Definition at line 584 of file `IpIteratesVector.hpp`.

```
6.76.3.4  virtual Vector* Ipopt::IteratesVectorSpace::MakeNew ( ) const [inline], [virtual]
```

This method creates a new vector (and allocates space in all the contained vectors).

This is really only used for code that does not know what type of vector it is dealing with - for example, this method is called from [Vector::MakeNew\(\)](#)

Reimplemented from [Ipopt::CompoundVectorSpace](#).

Definition at line 594 of file `IpIteratesVector.hpp`.

```
6.76.3.5  virtual void Ipopt::IteratesVectorSpace::SetCompSpace ( Index icom, const VectorSpace & vec_space )
        [inline], [virtual]
```

This method hides the [CompoundVectorSpace::SetCompSpace](#) method since the components of the `Iterates` are fixed at construction.

Reimplemented from [Ipopt::CompoundVectorSpace](#).

Definition at line 604 of file `IpIteratesVector.hpp`.

```
6.76.3.6  IteratesVectorSpace& Ipopt::IteratesVectorSpace::operator= ( const IteratesVectorSpace & ) [private]
```

Overloaded Equals Operator.

6.76.4 Member Data Documentation

```
6.76.4.1  SmartPtr<const VectorSpace> Ipopt::IteratesVectorSpace::x_space_ [private]
```

Contained Spaces.

Definition at line 627 of file `IpIteratesVector.hpp`.

```
6.76.4.2  SmartPtr<const VectorSpace> Ipopt::IteratesVectorSpace::s_space_ [private]
```

Definition at line 628 of file `IpIteratesVector.hpp`.

6.76.4.3 **SmartPointer<const VectorSpace>** Ipopt::IteratesVectorSpace::y_c_space_ [private]

Definition at line 629 of file IpIteratesVector.hpp.

6.76.4.4 **SmartPointer<const VectorSpace>** Ipopt::IteratesVectorSpace::y_d_space_ [private]

Definition at line 630 of file IpIteratesVector.hpp.

6.76.4.5 **SmartPointer<const VectorSpace>** Ipopt::IteratesVectorSpace::z_L_space_ [private]

Definition at line 631 of file IpIteratesVector.hpp.

6.76.4.6 **SmartPointer<const VectorSpace>** Ipopt::IteratesVectorSpace::z_U_space_ [private]

Definition at line 632 of file IpIteratesVector.hpp.

6.76.4.7 **SmartPointer<const VectorSpace>** Ipopt::IteratesVectorSpace::v_L_space_ [private]

Definition at line 633 of file IpIteratesVector.hpp.

6.76.4.8 **SmartPointer<const VectorSpace>** Ipopt::IteratesVectorSpace::v_U_space_ [private]

Definition at line 634 of file IpIteratesVector.hpp.

The documentation for this class was generated from the following file:

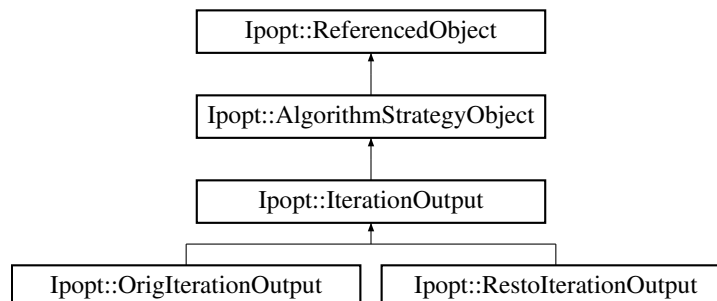
- [Algorithm/IpIteratesVector.hpp](#)

6.77 Ipopt::IterationOutput Class Reference

Base class for objects that do the output summary per iteration.

```
#include <IpIterationOutput.hpp>
```

Inheritance diagram for Ipopt::IterationOutput:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)
- virtual void [WriteOutput](#) ()=0
Method to do all the summary output per iteration.

Constructors/Destructors

- `IterationOutput ()`
Default Constructor.
- `virtual ~IterationOutput ()`
Default destructor.

Protected Types

- `enum InfPrOutput { INTERNAL =0, ORIGINAL }`
enumeration for different inf_pr output options

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `IterationOutput (const IterationOutput &)`
Copy Constructor.
- `void operator= (const IterationOutput &)`
Overloaded Equals Operator.

Additional Inherited Members

6.77.1 Detailed Description

Base class for objects that do the output summary per iteration.

Definition at line 22 of file `IterationOutput.hpp`.

6.77.2 Member Enumeration Documentation

6.77.2.1 `enum Ipopt::IterationOutput::InfPrOutput` `[protected]`

enumeration for different inf_pr output options

Enumerator

INTERNAL

ORIGINAL

Definition at line 47 of file `IterationOutput.hpp`.

6.77.3 Constructor & Destructor Documentation

6.77.3.1 `Ipopt::IterationOutput::IterationOutput ()` `[inline]`

Default Constructor.

Definition at line 28 of file `IterationOutput.hpp`.

6.77.3.2 `virtual Ipopt::IterationOutput::~~IterationOutput () [inline],[virtual]`

Default destructor.

Definition at line 32 of file `IpIterationOutput.hpp`.

6.77.3.3 `Ipopt::IterationOutput::IterationOutput (const IterationOutput &) [private]`

Copy Constructor.

6.77.4 Member Function Documentation

6.77.4.1 `virtual bool Ipopt::IterationOutput::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::RestIterationOutput](#), and [Ipopt::OrigIterationOutput](#).

6.77.4.2 `virtual void Ipopt::IterationOutput::WriteOutput () [pure virtual]`

Method to do all the summary output per iteration.

This include the one-line summary output as well as writing the details about the iterates if desired

Implemented in [Ipopt::RestIterationOutput](#), and [Ipopt::OrigIterationOutput](#).

6.77.4.3 `void Ipopt::IterationOutput::operator= (const IterationOutput &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

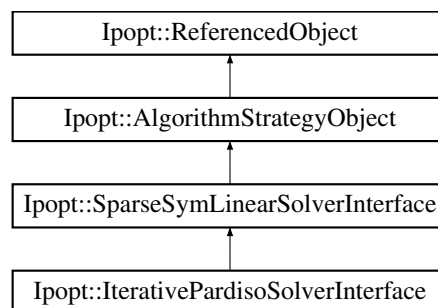
- [Algorithm/IpIterationOutput.hpp](#)

6.78 Ipopt::IterativePardisoSolverInterface Class Reference

Interface to the linear solver Pardiso, derived from [SparseSymLinearSolverInterface](#).

```
#include <IpIterativePardisoSolverInterface.hpp>
```

Inheritance diagram for `Ipopt::IterativePardisoSolverInterface`:



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Constructor/Destructor

- [IterativePardisoSolverInterface](#) ([IterativeSolverTerminationTester](#) &normal_tester, [IterativeSolverTerminationTester](#) &pd_tester)
Constructor.
- virtual [~IterativePardisoSolverInterface](#) ()
Destructor.

Methods for requesting solution of the linear system.

- virtual [ESymSolverStatus](#) [InitializeStructure](#) ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *ia, const [Index](#) *ja)
Method for initializing internal structures.
- virtual double * [GetValuesArrayPtr](#) ()
Method returning an internal array into which the nonzero elements are to be stored.
- virtual [ESymSolverStatus](#) [MultiSolve](#) (bool new_matrix, const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)
Solve operation for multiple right hand sides.
- virtual [Index](#) [NumberOfNegEVals](#) () const
Number of negative eigenvalues detected during last factorization.
- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat](#) [MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for lpoptType.

Private Member Functions

- [InexactData](#) & [InexData](#) ()
Method to easily access Inexact data.
- [InexactCq](#) & [InexCq](#) ()
Method to easily access Inexact calculated quantities.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IterativePardisoSolverInterface](#) ()

Default Constructor.

- [IterativePardisoSolverInterface](#) (const [IterativePardisoSolverInterface](#) &)

Copy Constructor.

- void [operator=](#) (const [IterativePardisoSolverInterface](#) &)

Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus SymbolicFactorization](#) (const [Index](#) *ia, const [Index](#) *ja)

Call Pardiso to do the analysis phase.

- [ESymSolverStatus Factorization](#) (const [Index](#) *ia, const [Index](#) *ja, bool check_NegEVals, [Index](#) numberOfNegEVals)

Call Pardiso to factorize the [Matrix](#).

- [ESymSolverStatus Solve](#) (const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals)

Call Pardiso to do the Solve.

Private Attributes

- [Number](#) [decr_factor_](#)

Decrease factor for dropping tolerances.

- [SmartPtr](#)

< [IterativeSolverTerminationTester](#) > [normal_tester_](#)

Termination tester for normal step computation.

- [SmartPtr](#)

< [IterativeSolverTerminationTester](#) > [pd_tester_](#)

Termination tester for primal-dual step computation.

Information about the matrix

- [Index](#) [dim_](#)

Number of rows and columns of the matrix.

- [Index](#) [nonzeros_](#)

Number of nonzeros of the matrix in triplet representation.

- double * [a_](#)

Array for storing the values of the matrix.

Information about most recent factorization/solve

- [Index](#) [negevals_](#)

Number of negative eigenvalues.

- [Index](#) [pardiso_max_iter_](#)

Options for the preconditioner.

- [Number](#) [pardiso_iter_relative_tol_](#)

- [Index](#) [pardiso_iter_coarse_size_](#)

- [Index](#) [pardiso_iter_max_levels_](#)

- [Number](#) [pardiso_iter_dropping_factor_](#)

- [Number](#) [pardiso_iter_dropping_schur_](#)

- [Index](#) [pardiso_iter_max_row_fill_](#)

- [Number](#) [pardiso_iter_inverse_norm_factor_](#)

- [Index](#) [normal_pardiso_max_iter_](#)

- [Number normal_pardiso_iter_relative_tol_](#)
- [Index normal_pardiso_iter_coarse_size_](#)
- [Index normal_pardiso_iter_max_levels_](#)
- [Number normal_pardiso_iter_dropping_factor_](#)
- [Number normal_pardiso_iter_dropping_schur_](#)
- [Index normal_pardiso_iter_max_row_fill_](#)
- [Number normal_pardiso_iter_inverse_norm_factor_](#)
- [Number pardiso_iter_dropping_factor_used_](#)
Actually used dropping tolerances.
- [Number pardiso_iter_dropping_schur_used_](#)
- [Number normal_pardiso_iter_dropping_factor_used_](#)
- [Number normal_pardiso_iter_dropping_schur_used_](#)

Initialization flags

- [bool initialized_](#)
Flag indicating if internal data is initialized.

Solver specific information

- [void ** PT_](#)
Internal data address pointers.
- [ipfint MAXFCT_](#)
Maximal number of factors with identical nonzero structure.
- [ipfint MNUM_](#)
Actual matrix for the solution phase.
- [ipfint MTYPE_](#)
Matrix type; real and symmetric indefinite.
- [ipfint * IPARM_](#)
Parameter and info array for Pardiso.
- [double * DPARM_](#)
Parameter and info array for Pardiso.
- [ipfint MSGVLV_](#)
Message level.

Some counters for debugging

- [Index debug_last_iter_](#)
- [Index debug_cnt_](#)

Solver specific options

- [enum PardisoMatchingStrategy { COMPLETE, COMPLETE2x2, CONSTRAINT }](#)
Type for mathcing strategies.
- [PardisoMatchingStrategy match_strat_](#)
Option that controls the matching strategy.
- [bool have_symbolic_factorization_](#)
Flag indicating if symbolic factorization has already been performed.
- [bool pardiso_redo_symbolic_fact_only_if_inertia_wrong_](#)
Flag indicating whether the symbolic factorization should only be done after perturbed elements, if the inertia was wrong.

- bool [pardiso_repeated_perturbation_means_singular_](#)
Flag indicating whether repeated perturbed elements even after a new symbolic factorization should be interpreted as a singular matrix.
- bool [skip_inertia_check_](#)
Flag indicating if the inertia is always assumed to be correct.
- Index [pardiso_max_droptol_corrections_](#)
Maximal number of decreases of drop tolerance during one solve.

Additional Inherited Members

6.78.1 Detailed Description

Interface to the linear solver Pardiso, derived from [SparseSymLinearSolverInterface](#).

For details, see description of [SparseSymLinearSolverInterface](#) base class.

Definition at line 25 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.2 Member Enumeration Documentation

6.78.2.1 enum `Ipopt::IterativePardisoSolverInterface::PardisoMatchingStrategy` `[private]`

Type for matching strategies.

Enumerator

COMPLETE
COMPLETE2x2
CONSTRAINT

Definition at line 136 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.3 Constructor & Destructor Documentation

6.78.3.1 `Ipopt::IterativePardisoSolverInterface::IterativePardisoSolverInterface (IterativeSolverTerminationTester & normal_tester, IterativeSolverTerminationTester & pd_tester)`

Constructor.

6.78.3.2 `virtual Ipopt::IterativePardisoSolverInterface::~~IterativePardisoSolverInterface ()` `[virtual]`

Destructor.

6.78.3.3 `Ipopt::IterativePardisoSolverInterface::IterativePardisoSolverInterface ()` `[private]`

Default Constructor.

6.78.3.4 `Ipopt::IterativePardisoSolverInterface::IterativePardisoSolverInterface (const IterativePardisoSolverInterface &)` `[private]`

Copy Constructor.

6.78.4 Member Function Documentation

6.78.4.1 `bool lpopt::IterativePardisoSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix)` `[virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.78.4.2 `virtual ESymSolverStatus lpopt::IterativePardisoSolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * ia, const Index * ja)` `[virtual]`

Method for initializing internal structures.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.78.4.3 `virtual double* lpopt::IterativePardisoSolverInterface::GetValuesArrayPtr ()` `[virtual]`

Method returning an internal array into which the nonzero elements are to be stored.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.78.4.4 `virtual ESymSolverStatus lpopt::IterativePardisoSolverInterface::MultiSolve (bool new_matrix, const Index * ia, const Index * ja, Index nrhs, double * rhs_vals, bool check_NegEVals, Index numberOfNegEVals)` `[virtual]`

Solve operation for multiple right hand sides.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.78.4.5 `virtual Index lpopt::IterativePardisoSolverInterface::NumberOfNegEVals () const` `[virtual]`

Number of negative eigenvalues detected during last factorization.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.78.4.6 `virtual bool lpopt::IterativePardisoSolverInterface::IncreaseQuality ()` `[virtual]`

Request to increase quality of solution for next solve.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.78.4.7 `virtual bool lpopt::IterativePardisoSolverInterface::ProvidesInertia () const` `[inline],[virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [lpopt::SparseSymLinearSolverInterface](#).

Definition at line 78 of file `lpIterativePardisoSolverInterface.hpp`.

6.78.4.8 `EMatrixFormat lpopt::IterativePardisoSolverInterface::MatrixFormat () const` `[inline],[virtual]`

Query of requested matrix type that the linear solver understands.

Implements [lpopt::SparseSymLinearSolverInterface](#).

Definition at line 85 of file `lpIterativePardisoSolverInterface.hpp`.

6.78.4.9 `static void Ipopt::IterativePardisoSolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > roptions)`
[static]

Methods for IpoptType.

6.78.4.10 `void Ipopt::IterativePardisoSolverInterface::operator= (const IterativePardisoSolverInterface &)` [private]

Overloaded Equals Operator.

6.78.4.11 `ESymSolverStatus Ipopt::IterativePardisoSolverInterface::SymbolicFactorization (const Index * ia, const Index * ja)` [private]

Call Pardiso to do the analysis phase.

6.78.4.12 `ESymSolverStatus Ipopt::IterativePardisoSolverInterface::Factorization (const Index * ia, const Index * ja, bool check_NegEVals, Index numberOfNegEVals)` [private]

Call Pardiso to factorize the [Matrix](#).

6.78.4.13 `ESymSolverStatus Ipopt::IterativePardisoSolverInterface::Solve (const Index * ia, const Index * ja, Index nrhs, double * rhs_vals)` [private]

Call Pardiso to do the Solve.

6.78.4.14 `InexactData& Ipopt::IterativePardisoSolverInterface::InexData ()` [inline],[private]

Method to easily access Inexact data.

Definition at line 248 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.4.15 `InexactCq& Ipopt::IterativePardisoSolverInterface::InexCq ()` [inline],[private]

Method to easily access Inexact calculated quantities.

Definition at line 257 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5 Member Data Documentation

6.78.5.1 `Index Ipopt::IterativePardisoSolverInterface::dim_` [private]

Number of rows and columns of the matrix.

Definition at line 118 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.2 `Index Ipopt::IterativePardisoSolverInterface::nonzeros_` [private]

Number of nonzeros of the matrix in triplet representation.

Definition at line 121 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.3 `double* Ipopt::IterativePardisoSolverInterface::a_` [private]

Array for storing the values of the matrix.

Definition at line 124 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.4 `Index Ipopt::IterativePardisoSolverInterface::negevals_` [private]

Number of negative eigenvalues.

Definition at line 130 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.5 **PardisoMatchingStrategy** `lpopt::IterativePardisoSolverInterface::match_strat_` [private]

Option that controls the matching strategy.

Definition at line 143 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.6 **bool** `lpopt::IterativePardisoSolverInterface::have_symbolic_factorization_` [private]

Flag indicating if symbolic factorization has already been performed.

Definition at line 146 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.7 **bool** `lpopt::IterativePardisoSolverInterface::pardiso_redo_symbolic_fact_only_if_inertia_wrong_` [private]

Flag indicating whether the symbolic factorization should only be done after perturbed elements, if the inertia was wrong.

Definition at line 149 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.8 **bool** `lpopt::IterativePardisoSolverInterface::pardiso_repeated_perturbation_means_singular_` [private]

Flag indicating whether repeated perturbed elements even after a new symbolic factorization should be interpreted as a singular matrix.

Definition at line 153 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.9 **bool** `lpopt::IterativePardisoSolverInterface::skip_inertia_check_` [private]

Flag indicating if the inertia is always assumed to be correct.

Definition at line 156 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.10 **Index** `lpopt::IterativePardisoSolverInterface::pardiso_max_droptol_corrections_` [private]

Maximal number of decreases of drop tolerance during one solve.

Definition at line 158 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.11 **Index** `lpopt::IterativePardisoSolverInterface::pardiso_max_iter_` [private]

Options for the preconditioner.

Definition at line 163 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.12 **Number** `lpopt::IterativePardisoSolverInterface::pardiso_iter_relative_tol_` [private]

Definition at line 164 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.13 **Index** `lpopt::IterativePardisoSolverInterface::pardiso_iter_coarse_size_` [private]

Definition at line 165 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.14 **Index** `lpopt::IterativePardisoSolverInterface::pardiso_iter_max_levels_` [private]

Definition at line 166 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.15 **Number** `lpopt::IterativePardisoSolverInterface::pardiso_iter_dropping_factor_` [private]

Definition at line 167 of file `lpliterativePardisoSolverInterface.hpp`.

6.78.5.16 **Number** Ipopt::IterativePardisoSolverInterface::pardiso_iter_dropping_schur_ [private]

Definition at line 168 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.17 **Index** Ipopt::IterativePardisoSolverInterface::pardiso_iter_max_row_fill_ [private]

Definition at line 169 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.18 **Number** Ipopt::IterativePardisoSolverInterface::pardiso_iter_inverse_norm_factor_ [private]

Definition at line 170 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.19 **Index** Ipopt::IterativePardisoSolverInterface::normal_pardiso_max_iter_ [private]

Definition at line 172 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.20 **Number** Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_relative_tol_ [private]

Definition at line 173 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.21 **Index** Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_coarse_size_ [private]

Definition at line 174 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.22 **Index** Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_max_levels_ [private]

Definition at line 175 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.23 **Number** Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_dropping_factor_ [private]

Definition at line 176 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.24 **Number** Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_dropping_schur_ [private]

Definition at line 177 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.25 **Index** Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_max_row_fill_ [private]

Definition at line 178 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.26 **Number** Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_inverse_norm_factor_ [private]

Definition at line 179 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.27 **Number** Ipopt::IterativePardisoSolverInterface::decr_factor_ [private]

Decrease factor for dropping tolerances.

Definition at line 183 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.28 **Number** Ipopt::IterativePardisoSolverInterface::pardiso_iter_dropping_factor_used_ [private]

Actually used dropping tolerances.

Definition at line 187 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.29 **Number** Ipopt::IterativePardisoSolverInterface::pardiso_iter_dropping_schur_used_ [private]

Definition at line 188 of file IpIterativePardisoSolverInterface.hpp.

6.78.5.30 `Number Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_dropping_factor_used_ [private]`

Definition at line 189 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.31 `Number Ipopt::IterativePardisoSolverInterface::normal_pardiso_iter_dropping_schur_used_ [private]`

Definition at line 190 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.32 `bool Ipopt::IterativePardisoSolverInterface::initialized_ [private]`

Flag indicating if internal data is initialized.

For initialization, this object needs to have seen a matrix

Definition at line 197 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.33 `void** Ipopt::IterativePardisoSolverInterface::PT_ [private]`

Internal data address pointers.

Definition at line 203 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.34 `ipfint Ipopt::IterativePardisoSolverInterface::MAXFCT_ [private]`

Maximal number of factors with identical nonzero structure.

Here, we only store one factorization. Is always 1.

Definition at line 206 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.35 `ipfint Ipopt::IterativePardisoSolverInterface::MNUM_ [private]`

Actual matrix for the solution phase.

Is always 1.

Definition at line 208 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.36 `ipfint Ipopt::IterativePardisoSolverInterface::MTYPE_ [private]`

[Matrix](#) type; real and symmetric indefinite.

Is always -2.

Definition at line 210 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.37 `ipfint* Ipopt::IterativePardisoSolverInterface::IPARM_ [private]`

Parameter and info array for Pardiso.

Definition at line 212 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.38 `double* Ipopt::IterativePardisoSolverInterface::DPARM_ [private]`

Parameter and info array for Pardiso.

Definition at line 214 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.39 `ipfint Ipopt::IterativePardisoSolverInterface::MSGVLVL_ [private]`

Message level.

Definition at line 216 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.40 Index `Ipopt::IterativePardisoSolverInterface::debug_last_iter_` `[private]`

Definition at line 221 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.41 Index `Ipopt::IterativePardisoSolverInterface::debug_cnt_` `[private]`

Definition at line 222 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.42 SmartPtr<IterativeSolverTerminationTester> `Ipopt::IterativePardisoSolverInterface::normal_tester_` `[private]`

Termination tester for normal step computation.

Definition at line 266 of file `IpIterativePardisoSolverInterface.hpp`.

6.78.5.43 SmartPtr<IterativeSolverTerminationTester> `Ipopt::IterativePardisoSolverInterface::pd_tester_` `[private]`

Termination tester for primal-dual step computation.

Definition at line 269 of file `IpIterativePardisoSolverInterface.hpp`.

The documentation for this class was generated from the following file:

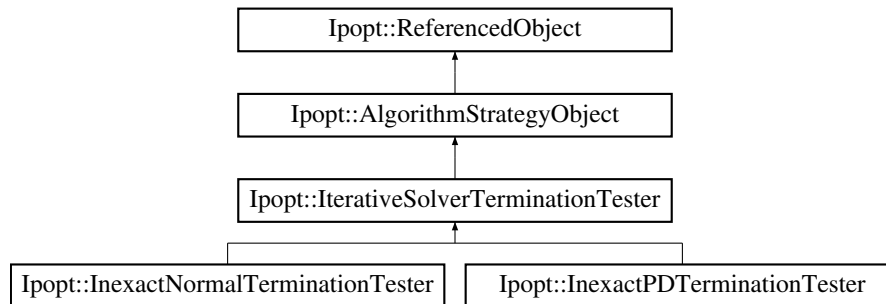
- Algorithm/Inexact/[IpIterativePardisoSolverInterface.hpp](#)

6.79 Ipopt::IterativeSolverTerminationTester Class Reference

This base class is for the termination tests for the iterative linear solver in the inexact version of [Ipopt](#).

```
#include <IpIterativeSolverTerminationTester.hpp>
```

Inheritance diagram for `Ipopt::IterativeSolverTerminationTester`:

**Public Types**

- enum [ETerminationTest](#) {
[CONTINUE](#), [TEST_1_SATISFIED](#), [TEST_2_SATISFIED](#), [TEST_3_SATISFIED](#),
[MODIFY_HESSIAN](#), [OTHER_SATISFIED](#) }

Enum to report result of termination test.

Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
Implementation of the initialization method that has to be overloaded by for each derived class.

- virtual bool `InitializeSolve` ()=0
Method for initializing for the next iterative solve.
- virtual `ETerminationTest TestTermination` (Index ndim, const `Number` *sol, const `Number` *resid, Index iter, `Number` norm2_rhs)=0
This method checks if the current solution of the iterative linear solver is good enough (by returning the corresponding satisfied termination test), or if the Hessian should be modified.
- virtual void `Clear` ()=0
This method can be called after the Solve is over and we can delete anything that has been allocated to free memory.
- const `Journalist` & `GetJnlst` () const
An easy way to get the journalist if accessed from the outside.
- virtual Index `GetSolverIterations` () const =0
Return the number of iterative solver iteration from the most recent solve.

/Destructor

- `IterativeSolverTerminationTester` ()
Default constructor.
- virtual `~IterativeSolverTerminationTester` ()
Default destructor.

Protected Member Functions

- void `GetVectors` (Index ndim, const `Number` *array, `SmartPtr`< const `Vector` > &comp_x, `SmartPtr`< const `Vector` > &comp_s, `SmartPtr`< const `Vector` > &comp_c, `SmartPtr`< const `Vector` > &comp_d)
Method for copying a long augmented system array into Vectors in `lpopt` notation.
- `InexactData` & `InexData` ()
Method to easily access Inexact data.
- `InexactCq` & `InexCq` ()
Method to easily access Inexact calculated quantities.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `IterativeSolverTerminationTester` & `operator=` (const `IterativeSolverTerminationTester` &)
Overloaded Equals Operator.

6.79.1 Detailed Description

This base class is for the termination tests for the iterative linear solver in the inexact version of `lpopt`.

Definition at line 21 of file `lpIterativeSolverTerminationTester.hpp`.

6.79.2 Member Enumeration Documentation

6.79.2.1 enum Ipopt::IterativeSolverTerminationTester::ETerminationTest

Enum to report result of termination test.

Enumerator

- CONTINUE** The current solution is not yet good enough.
- TEST_1_SATISFIED** Termination Test 1 is satisfied.
- TEST_2_SATISFIED** Termination Test 2 is satisfied.
- TEST_3_SATISFIED** Termination Test 3 is satisfied.
- MODIFY_HESSIAN** Hessian matrix should be modified.
- OTHER_SATISFIED** Some other termination criterion satisfied.

Definition at line 25 of file IpIterativeSolverTerminationTester.hpp.

6.79.3 Constructor & Destructor Documentation

6.79.3.1 Ipopt::IterativeSolverTerminationTester::IterativeSolverTerminationTester () [inline]

Default constructor.

Definition at line 44 of file IpIterativeSolverTerminationTester.hpp.

6.79.3.2 virtual Ipopt::IterativeSolverTerminationTester::~~IterativeSolverTerminationTester () [inline], [virtual]

Default destructor.

Definition at line 48 of file IpIterativeSolverTerminationTester.hpp.

6.79.4 Member Function Documentation

6.79.4.1 virtual bool Ipopt::IterativeSolverTerminationTester::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]

Implementation of the initialization method that has to be overloaded by for each derived class.

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::InexactNormalTerminationTester](#), and [Ipopt::InexactPDTerminationTester](#).

6.79.4.2 virtual bool Ipopt::IterativeSolverTerminationTester::InitializeSolve () [pure virtual]

Method for initializing for the next iterative solve.

This must be call before the test methods are called.

Implemented in [Ipopt::InexactNormalTerminationTester](#), and [Ipopt::InexactPDTerminationTester](#).

6.79.4.3 virtual ETerminationTest Ipopt::IterativeSolverTerminationTester::TestTermination (Index ndim, const Number * sol, const Number * resid, Index iter, Number norm2_rhs) [pure virtual]

This method checks if the current soltion of the iterative linear solver is good enough (by returning the corresponding satisfied termination test), or if the Hessian should be modified.

The input is the dimension of the augmented system, the current solution vector of the augmented system, the current residual vector.

Implemented in [lpopt::InexactNormalTerminationTester](#), and [lpopt::InexactPDTerminationTester](#).

6.79.4.4 `virtual void lpopt::IterativeSolverTerminationTester::Clear () [pure virtual]`

This method can be called after the Solve is over and we can delete anything that has been allocated to free memory.

Implemented in [lpopt::InexactNormalTerminationTester](#), and [lpopt::InexactPDTerminationTester](#).

6.79.4.5 `const Journalist& lpopt::IterativeSolverTerminationTester::GetJnlst () const [inline]`

An easy way to get the journalist if accessed from the outside.

Definition at line 76 of file `lpIterativeSolverTerminationTester.hpp`.

6.79.4.6 `virtual Index lpopt::IterativeSolverTerminationTester::GetSolverIterations () const [pure virtual]`

Return the number of iterative solver iteration from the most recent solve.

Implemented in [lpopt::InexactNormalTerminationTester](#), and [lpopt::InexactPDTerminationTester](#).

6.79.4.7 `void lpopt::IterativeSolverTerminationTester::GetVectors (Index ndim, const Number * array, SmartPtr< const Vector > & comp_x, SmartPtr< const Vector > & comp_s, SmartPtr< const Vector > & comp_c, SmartPtr< const Vector > & comp_d) [protected]`

Method for copying a long augmented system array into Vectors in [lpopt](#) notation.

6.79.4.8 `InexactData& lpopt::IterativeSolverTerminationTester::InexData () [inline], [protected]`

Method to easily access Inexact data.

Definition at line 95 of file `lpIterativeSolverTerminationTester.hpp`.

6.79.4.9 `InexactCq& lpopt::IterativeSolverTerminationTester::InexCq () [inline], [protected]`

Method to easily access Inexact calculated quantities.

Definition at line 104 of file `lpIterativeSolverTerminationTester.hpp`.

6.79.4.10 `IterativeSolverTerminationTester& lpopt::IterativeSolverTerminationTester::operator= (const IterativeSolverTerminationTester &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

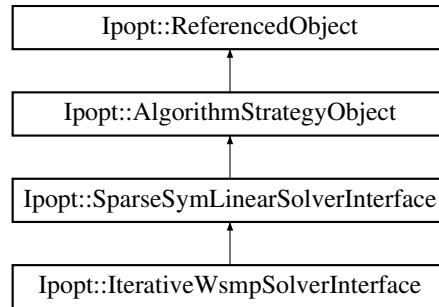
- Algorithm/Inexact/[lpIterativeSolverTerminationTester.hpp](#)

6.80 lpopt::IterativeWsmSolverInterface Class Reference

Interface to the linear solver WISMP, derived from [SparseSymLinearSolverInterface](#).

```
#include <lpIterativeWsmSolverInterface.hpp>
```

Inheritance diagram for `lpopt::IterativeWsmSolverInterface`:



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Constructor/Destructor

- [IterativeWsmSolverInterface](#) ()
Constructor.
- virtual [~IterativeWsmSolverInterface](#) ()
Destructor.

Methods for requesting solution of the linear system.

- virtual [ESymSolverStatus](#) [InitializeStructure](#) ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *ia, const [Index](#) *ja)
Method for initializing internal stuctures.
- virtual double * [GetValuesArrayPtr](#) ()
Method returing an internal array into which the nonzero elements are to be stored.
- virtual [ESymSolverStatus](#) [MultiSolve](#) (bool new_matrix, const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)
Solve operation for multiple right hand sides.
- virtual [Index](#) [NumberOfNegEVals](#) () const
Number of negative eigenvalues detected during last factorization.
- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat](#) [MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [IterativeWsmSolverInterface](#) (const [IterativeWsmSolverInterface](#) &)
Copy Constructor.
- void [operator=](#) (const [IterativeWsmSolverInterface](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus SymbolicFactorization](#) (const [Index](#) *ia, const [Index](#) *ja)
Call Wsm to do the analysis phase.
- [ESymSolverStatus InternalSymFact](#) (const [Index](#) *ia, const [Index](#) *ja)
Call Wsm to really do the analysis phase.
- [ESymSolverStatus Factorization](#) (const [Index](#) *ia, const [Index](#) *ja, bool check_NegEVals, [Index](#) numberOfNegEVals)
Call Wsm to factorize the [Matrix](#).
- [ESymSolverStatus Solve](#) (const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals)
Call Wsm to do the Solve.

Private Attributes

- [Index matrix_file_number_](#)
Counter for matrix file numbers.

Information about the matrix

- [Index dim_](#)
Number of rows and columns of the matrix.
- double * [a_](#)
Array for storing the values of the matrix.

Solver specific options

- [Index wsm_num_threads_](#)
Option that controls the matching strategy.
- [Number wsm_pivotol_](#)
Pivot tolerance.
- [Number wsm_pivotolmax_](#)
Maximal pivot tolerance.
- [Index wsm_scaling_](#)
Indicating which of WSMP's scaling methods should be used.
- [Index wsm_write_matrix_iteration_](#)
iteration number in which matrices are to be written out
- [Number wsm_inexact_droptol_](#)
- [Number wsm_inexact_fillin_limit_](#)

Initialization flags

- bool [initialized_](#)
Flag indicating if internal data is initialized.
- bool [pivtol_changed_](#)
Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed.
- bool [have_symbolic_factorization_](#)
Flag indicating whether symbolic factorization and order has already been performed.

Solver specific information

- [ipfint](#) * [IPARM_](#)
Integer parameter array for WISMP.
- [double](#) * [DPARM_](#)
Double precision parameter array for WISMP.

Additional Inherited Members

6.80.1 Detailed Description

Interface to the linear solver WISMP, derived from [SparseSymLinearSolverInterface](#).

For details, see description of [SparseSymLinearSolverInterface](#) base class.

Definition at line 23 of file `IpIterativeWsmSolverInterface.hpp`.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 Ipopt::IterativeWsmSolverInterface::IterativeWsmSolverInterface ()

Constructor.

6.80.2.2 virtual Ipopt::IterativeWsmSolverInterface::~~IterativeWsmSolverInterface () [virtual]

Destructor.

6.80.2.3 Ipopt::IterativeWsmSolverInterface::IterativeWsmSolverInterface (const IterativeWsmSolverInterface &) [private]

Copy Constructor.

6.80.3 Member Function Documentation

6.80.3.1 bool Ipopt::IterativeWsmSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.80.3.2 virtual ESymSolverStatus Ipopt::IterativeWsmSolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * ia, const Index * ja) [virtual]

Method for initializing internal structures.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.80.3.3 `virtual double* lpopt::IterativeWsmSolverInterface::GetValuesArrayPtr () [virtual]`

Method returning an internal array into which the nonzero elements are to be stored.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.80.3.4 `virtual ESymSolverStatus lpopt::IterativeWsmSolverInterface::MultiSolve (bool new_matrix, const Index * ia, const Index * ja, Index nrhs, double * rhs_vals, bool check_NegEVals, Index numberOfNegEVals) [virtual]`

Solve operation for multiple right hand sides.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.80.3.5 `virtual Index lpopt::IterativeWsmSolverInterface::NumberOfNegEVals () const [virtual]`

Number of negative eigenvalues detected during last factorization.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.80.3.6 `virtual bool lpopt::IterativeWsmSolverInterface::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.80.3.7 `virtual bool lpopt::IterativeWsmSolverInterface::ProvidesInertia () const [inline],[virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [lpopt::SparseSymLinearSolverInterface](#).

Definition at line 75 of file `lpopt::IterativeWsmSolverInterface.hpp`.

6.80.3.8 `EMatrixFormat lpopt::IterativeWsmSolverInterface::MatrixFormat () const [inline],[virtual]`

Query of requested matrix type that the linear solver understands.

Implements [lpopt::SparseSymLinearSolverInterface](#).

Definition at line 82 of file `lpopt::IterativeWsmSolverInterface.hpp`.

6.80.3.9 `static void lpopt::IterativeWsmSolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > options) [static]`

Methods for `lpoptType`.

6.80.3.10 `void lpopt::IterativeWsmSolverInterface::operator= (const IterativeWsmSolverInterface &) [private]`

Overloaded Equals Operator.

6.80.3.11 `ESymSolverStatus lpopt::IterativeWsmSolverInterface::SymbolicFactorization (const Index * ia, const Index * ja) [private]`

Call Wsm to do the analysis phase.

6.80.3.12 `ESymSolverStatus lpopt::IterativeWsmSolverInterface::InternalSymFact (const Index * ia, const Index * ja) [private]`

Call Wsm to really do the analysis phase.

6.80.3.13 **ESymSolverStatus** Ipopt::IterativeWsmSolverInterface::Factorization (const Index * *ia*, const Index * *ja*, bool *check_NegEvals*, Index *numberOfNegEvals*) [private]

Call Wsm to factorize the [Matrix](#).

6.80.3.14 **ESymSolverStatus** Ipopt::IterativeWsmSolverInterface::Solve (const Index * *ia*, const Index * *ja*, Index *nrhs*, double * *rhs_vals*) [private]

Call Wsmplx to do the Solve.

6.80.4 Member Data Documentation

6.80.4.1 **Index** Ipopt::IterativeWsmSolverInterface::dim_ [private]

Number of rows and columns of the matrix.

Definition at line 112 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.2 **double*** Ipopt::IterativeWsmSolverInterface::a_ [private]

Array for storing the values of the matrix.

Definition at line 115 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.3 **Index** Ipopt::IterativeWsmSolverInterface::wsm_num_threads_ [private]

Option that controls the matching strategy.

Definition at line 121 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.4 **Number** Ipopt::IterativeWsmSolverInterface::wsm_pivtol_ [private]

Pivot tolerance.

Definition at line 123 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.5 **Number** Ipopt::IterativeWsmSolverInterface::wsm_pivtolmax_ [private]

Maximal pivot tolerance.

Definition at line 125 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.6 **Index** Ipopt::IterativeWsmSolverInterface::wsm_scaling_ [private]

Indicating which of WSMP's scaling methods should be used.

Definition at line 127 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.7 **Index** Ipopt::IterativeWsmSolverInterface::wsm_write_matrix_iteration_ [private]

iteration number in which matrices are to be written out

Definition at line 129 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.8 **Number** Ipopt::IterativeWsmSolverInterface::wsm_inexact_droptol_ [private]

Definition at line 130 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.9 **Number** Ipopt::IterativeWsmSolverInterface::wsmp_inexact_fillin_limit_ [private]

Definition at line 131 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.10 **Index** Ipopt::IterativeWsmSolverInterface::matrix_file_number_ [private]

Counter for matrix file numbers.

Definition at line 135 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.11 **bool** Ipopt::IterativeWsmSolverInterface::initialized_ [private]

Flag indicating if internal data is initialized.

For initialization, this object needs to have seen a matrix

Definition at line 149 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.12 **bool** Ipopt::IterativeWsmSolverInterface::pivtol_changed_ [private]

Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed.

Definition at line 152 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.13 **bool** Ipopt::IterativeWsmSolverInterface::have_symbolic_factorization_ [private]

Flag indicating whether symbolic factorization and order has already been performed.

Definition at line 155 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.14 **ipfint*** Ipopt::IterativeWsmSolverInterface::IPARM_ [private]

Integer parameter array for WISMP.

Definition at line 161 of file IpIterativeWsmSolverInterface.hpp.

6.80.4.15 **double*** Ipopt::IterativeWsmSolverInterface::DPARM_ [private]

Double precision parameter array for WISMP.

Definition at line 163 of file IpIterativeWsmSolverInterface.hpp.

The documentation for this class was generated from the following file:

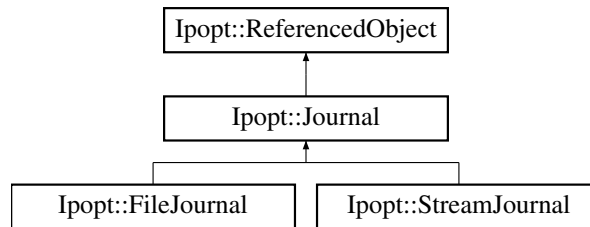
- Algorithm/LinearSolvers/[IpIterativeWsmSolverInterface.hpp](#)

6.81 Ipopt::Journal Class Reference

[Journal](#) class (part of the [Journalist](#) implementation.).

```
#include <IpJournalist.hpp>
```

Inheritance diagram for Ipopt::Journal:



Public Member Functions

- **Journal** (const std::string &name, [EJournalLevel](#) default_level)
Constructor.
- virtual **~Journal** ()
Destructor.
- virtual std::string **Name** ()
Get the name of the Journal.
- virtual void **SetPrintLevel** ([EJournalCategory](#) category, [EJournalLevel](#) level)
Set the print level for a particular category.
- virtual void **SetAllPrintLevels** ([EJournalLevel](#) level)
Set the print level for all category.

Journal Output Methods. These methods are called by the

[Journalist](#) who first checks if the output print level and category are acceptable.

Calling the Print methods explicitly (instead of through the [Journalist](#) will output the message regardless of print level and category. You should use the [Journalist](#) to print & flush instead

- virtual bool **IsAccepted** ([EJournalCategory](#) category, [EJournalLevel](#) level) const
Ask if a particular print level/category is accepted by the journal.
- virtual void **Print** ([EJournalCategory](#) category, [EJournalLevel](#) level, const char *str)
Print to the designated output location.
- virtual void **Printf** ([EJournalCategory](#) category, [EJournalLevel](#) level, const char *pformat, va_list ap)
Printf to the designated output location.
- virtual void **FlushBuffer** ()
Flush output buffer.

Protected Member Functions

Implementation version of Print methods. Derived classes

should overload the Impl methods.

- virtual void **PrintImpl** ([EJournalCategory](#) category, [EJournalLevel](#) level, const char *str)=0
Print to the designated output location.
- virtual void **PrintfImpl** ([EJournalCategory](#) category, [EJournalLevel](#) level, const char *pformat, va_list ap)=0
Printf to the designated output location.
- virtual void **FlushBufferImpl** ()=0
Flush output buffer.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [Journal](#) ()
Default Constructor.
- [Journal](#) (const [Journal](#) &)
Copy Constructor.
- void [operator=](#) (const [Journal](#) &)
Overloaded Equals Operator.

Private Attributes

- std::string [name_](#)
Name of the output location.
- [Index print_levels_](#) [[J_LAST_CATEGORY](#)]
vector of integers indicating the level for each category

6.81.1 Detailed Description

[Journal](#) class (part of the [Journalist](#) implementation.).

This class is the base class for all Journals. It controls the acceptance criteria for print statements etc. Derived classes like the [FileJournal](#) - output those messages to specific locations

Definition at line 273 of file [IpJournalist.hpp](#).

6.81.2 Constructor & Destructor Documentation

6.81.2.1 `Ipopt::Journal::Journal (const std::string & name, EJournalLevel default_level)`

Constructor.

6.81.2.2 `virtual Ipopt::Journal::~~Journal () [virtual]`

Destructor.

6.81.2.3 `Ipopt::Journal::Journal () [private]`

Default Constructor.

6.81.2.4 `Ipopt::Journal::Journal (const Journal &) [private]`

Copy Constructor.

6.81.3 Member Function Documentation

6.81.3.1 `virtual std::string Ipopt::Journal::Name () [virtual]`

Get the name of the [Journal](#).

6.81.3.2 `virtual void Ipopt::Journal::SetPrintLevel (EJournalCategory category, EJournalLevel level) [virtual]`

Set the print level for a particular category.

6.81.3.3 `virtual void Ipopt::Journal::SetAllPrintLevels (EJournalLevel level) [virtual]`

Set the print level for all category.

6.81.3.4 `virtual bool Ipopt::Journal::IsAccepted (EJournalCategory category, EJournalLevel level) const [virtual]`

Ask if a particular print level/category is accepted by the journal.

6.81.3.5 `virtual void Ipopt::Journal::Print (EJournalCategory category, EJournalLevel level, const char * str)
[inline],[virtual]`

Print to the designated output location.

Definition at line 311 of file IpJournalist.hpp.

6.81.3.6 `virtual void Ipopt::Journal::Printf (EJournalCategory category, EJournalLevel level, const char * pformat, va_list
ap) [inline],[virtual]`

Printf to the designated output location.

Definition at line 318 of file IpJournalist.hpp.

6.81.3.7 `virtual void Ipopt::Journal::FlushBuffer () [inline],[virtual]`

Flush output buffer.

Definition at line 325 of file IpJournalist.hpp.

6.81.3.8 `virtual void Ipopt::Journal::PrintImpl (EJournalCategory category, EJournalLevel level, const char * str)
[protected],[pure virtual]`

Print to the designated output location.

Implemented in [Ipopt::StreamJournal](#), and [Ipopt::FileJournal](#).

6.81.3.9 `virtual void Ipopt::Journal::PrintfImpl (EJournalCategory category, EJournalLevel level, const char * pformat,
va_list ap) [protected],[pure virtual]`

Printf to the designated output location.

Implemented in [Ipopt::StreamJournal](#), and [Ipopt::FileJournal](#).

6.81.3.10 `virtual void Ipopt::Journal::FlushBufferImpl () [protected],[pure virtual]`

Flush output buffer.

Implemented in [Ipopt::StreamJournal](#), and [Ipopt::FileJournal](#).

6.81.3.11 `void Ipopt::Journal::operator= (const Journal &) [private]`

Overloaded Equals Operator.

6.81.4 Member Data Documentation

6.81.4.1 `std::string Ipopt::Journal::name_ [private]`

Name of the output location.

Definition at line 368 of file `IpJournalist.hpp`.

6.81.4.2 `Index Ipopt::Journal::print_levels_[J_LAST_CATEGORY] [private]`

vector of integers indicating the level for each category

Definition at line 371 of file `IpJournalist.hpp`.

The documentation for this class was generated from the following file:

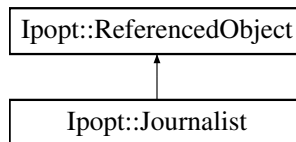
- [Common/IpJournalist.hpp](#)

6.82 Ipopt::Journalist Class Reference

Class responsible for all message output.

```
#include <IpJournalist.hpp>
```

Inheritance diagram for `Ipopt::Journalist`:



Public Member Functions

Constructor / Desructor.

- [Journalist](#) ()
Constructor.
- virtual [~Journalist](#) ()
Destructor...

Author Methods.

These methods are used by authoring code, or code that wants to report some information.

- virtual void [Printf](#) ([EJournalLevel](#) level, [EJournalCategory](#) category, const char *format,...) const
Method to print a formatted string.
- virtual void [PrintStringOverLines](#) ([EJournalLevel](#) level, [EJournalCategory](#) category, [Index](#) indent_spaces, [Index](#) max_length, const std::string &line) const
Method to print a long string including indentation.
- virtual void [PrintfIndented](#) ([EJournalLevel](#) level, [EJournalCategory](#) category, [Index](#) indent_level, const char *format,...) const
Method to print a formatted string with indentation.
- virtual void [VPrintf](#) ([EJournalLevel](#) level, [EJournalCategory](#) category, const char *pformat, va_list ap) const
Method to print a formatted string using the va_list argument.
- virtual void [VPrintfIndented](#) ([EJournalLevel](#) level, [EJournalCategory](#) category, [Index](#) indent_level, const char *pformat, va_list ap) const
Method to print a formatted string with indentation, using the va_list argument.

- virtual bool [ProduceOutput](#) ([EJournalLevel](#) level, [EJournalCategory](#) category) const
Method that returns true if there is a [Journal](#) that would write output for the given JournalLevel and JournalCategory.
- virtual void [FlushBuffer](#) () const
Method that flushes the current buffer for all Journalists.

Reader Methods.

These methods are used by the reader.

The reader will setup the journalist with each output file and the acceptance criteria for that file.

Use these methods to setup the journals (files or other output). These are the internal objects that keep track of the print levels for each category. Then use the internal [Journal](#) objects to set specific print levels for each category (or keep defaults).

- virtual bool [AddJournal](#) (const [SmartPtr](#)< [Journal](#) > jrnI)
Add a new journal.
- virtual [SmartPtr](#)< [Journal](#) > [AddFileJournal](#) (const std::string &location_name, const std::string &fname, [EJournalLevel](#) default_level=[J_WARNING](#))
Add a new [FileJournal](#).
- virtual [SmartPtr](#)< [Journal](#) > [GetJournal](#) (const std::string &location_name)
Get an existing journal.
- virtual void [DeleteAllJournals](#) ()
Delete all journals curenly known by the journalist.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [Journalist](#) (const [Journalist](#) &)
Copy Constructor.
- void [operator=](#) (const [Journalist](#) &)
Overloaded Equals Operator.

Private Attributes

- std::vector< [SmartPtr](#)< [Journal](#) > > [journals_](#)

6.82.1 Detailed Description

Class responsible for all message output.

This class is responsible for all messaging and output. The "printing" code or "author" should send ALL messages to the [Journalist](#), indicating an appropriate category and print level. The journalist then decides, based on reader specified acceptance criteria, which message is actually printed in which journals. This allows the printing code to send everything, while the "reader" can decide what they really want to see.

Authors: Authors use the Journals: You can add as many Journals as you like to the [Journalist](#) with the [AddJournal](#) or the [AddFileJournal](#) methods. Each one represents a different printing location (or file). Then, you can call the "print" methods of the [Journalist](#) to output information to each of the journals.

Acceptance Criteria: Each print message should be flagged appropriately with an [EJournalCategory](#) and [EJournalLevel](#).

The AddFileJournal method returns a pointer to the newly created [Journal](#) object (if successful) so you can set Acceptance criteria for that particular location.

Definition at line 134 of file IpJournalist.hpp.

6.82.2 Constructor & Destructor Documentation

6.82.2.1 Ipopt::Journalist::Journalist ()

Constructor.

6.82.2.2 virtual Ipopt::Journalist::~~Journalist () [virtual]

Destructor...

6.82.2.3 Ipopt::Journalist::Journalist (const Journalist &) [private]

Copy Constructor.

6.82.3 Member Function Documentation

6.82.3.1 virtual void Ipopt::Journalist::Printf (EJournalLevel *level*, EJournalCategory *category*, const char * *format*, ...) const [virtual]

Method to print a formatted string.

6.82.3.2 virtual void Ipopt::Journalist::PrintStringOverLines (EJournalLevel *level*, EJournalCategory *category*, Index *indent_spaces*, Index *max_length*, const std::string & *line*) const [virtual]

Method to print a long string including indentation.

The string is printed starting at the current position. If the position (counting started at the current position) exceeds max_length, a new line is inserted, and indent_spaces many spaces are printed before the string is continued. This is for example used during the printing of the option documentation.

6.82.3.3 virtual void Ipopt::Journalist::PrintfIndented (EJournalLevel *level*, EJournalCategory *category*, Index *indent_level*, const char * *format*, ...) const [virtual]

Method to print a formatted string with indentation.

6.82.3.4 virtual void Ipopt::Journalist::VPrintf (EJournalLevel *level*, EJournalCategory *category*, const char * *pformat*, va_list *ap*) const [virtual]

Method to print a formatted string using the va_list argument.

6.82.3.5 virtual void Ipopt::Journalist::VPrintfIndented (EJournalLevel *level*, EJournalCategory *category*, Index *indent_level*, const char * *pformat*, va_list *ap*) const [virtual]

Method to print a formatted string with indentation, using the va_list argument.

6.82.3.6 virtual bool Ipopt::Journalist::ProduceOutput (EJournalLevel *level*, EJournalCategory *category*) const [virtual]

Method that returns true if there is a [Journal](#) that would write output for the given JournalLevel and JournalCategory.

This is useful if expensive computation would be required for a particular output. The author code can check with this

method if the computations are indeed required.

6.82.3.7 `virtual void Ipopt::Journalist::FlushBuffer () const` `[virtual]`

Method that flushes the current buffer for all Journalists.

Calling this method after one optimization run helps to avoid cluttering output with that produced by other parts of the program (e.g. written in Fortran)

6.82.3.8 `virtual bool Ipopt::Journalist::AddJournal (const SmartPtr< Journal > jrn)` `[virtual]`

Add a new journal.

The `location_name` is a string identifier, which can be used to obtain the pointer to the new [Journal](#) at a later point using the `GetJournal` method. The `default_level` is used to initialize the * printing level for all categories.

6.82.3.9 `virtual SmartPtr<Journal> Ipopt::Journalist::AddFileJournal (const std::string & location_name, const std::string & fname, EJournalLevel default_level = J_WARNING)` `[virtual]`

Add a new [FileJournal](#).

`fname` is the name of the * file to which this [Journal](#) corresponds. Use `fname="stdout"` * for stdout, and use `fname="stderr"` for stderr. This method * returns the [Journal](#) pointer so you can set specific acceptance criteria. It returns NULL if there was a problem creating a new [Journal](#).

Parameters

<i>location_name</i>	journal identifier
<i>fname</i>	file name
<i>default_level</i>	default journal level

6.82.3.10 `virtual SmartPtr<Journal> Ipopt::Journalist::GetJournal (const std::string & location_name)` `[virtual]`

Get an existing journal.

You can use this method to change the acceptance criteria at runtime.

6.82.3.11 `virtual void Ipopt::Journalist::DeleteAllJournals ()` `[virtual]`

Delete all journals curently known by the journalist.

6.82.3.12 `void Ipopt::Journalist::operator= (const Journalist &)` `[private]`

Overloaded Equals Operator.

6.82.4 Member Data Documentation

6.82.4.1 `std::vector< SmartPtr<Journal> > Ipopt::Journalist::journals_` `[private]`

Definition at line 264 of file `IpJournalist.hpp`.

The documentation for this class was generated from the following file:

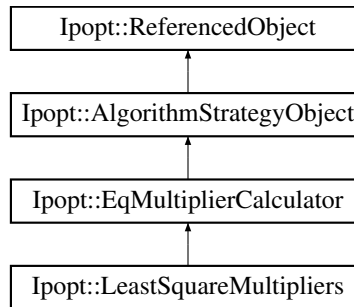
- [Common/IpJournalist.hpp](#)

6.83 Ipopt::LeastSquareMultipliers Class Reference

Class for calculator for the least-square equality constraint multipliers.

```
#include <IpLeastSquareMults.hpp>
```

Inheritance diagram for Ipopt::LeastSquareMultipliers:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [CalculateMultipliers](#) ([Vector](#) &y_c, [Vector](#) &y_d)
This method computes the least-square estimates for y_c and y_d at the current point.

Constructors/Destructors

- [LeastSquareMultipliers](#) ([AugSystemSolver](#) &augSysSolver)
Constructor.
- virtual [~LeastSquareMultipliers](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [LeastSquareMultipliers](#) ()
Default Constructor.
- [LeastSquareMultipliers](#) (const [LeastSquareMultipliers](#) &)
Copy Constructor.
- void [operator=](#) (const [LeastSquareMultipliers](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [AugSystemSolver](#) > [augsysolver_](#)
Pointer for the augmented system solver to be used for solving the linear system.

Additional Inherited Members

6.83.1 Detailed Description

Class for calculator for the least-square equality constraint multipliers.

The Calculate method of this class computes the least-square estimate for the `y_c` and `y_d` multipliers, based on the current values of the gradient of the Lagrangian.

Definition at line 23 of file `IpLeastSquareMults.hpp`.

6.83.2 Constructor & Destructor Documentation

6.83.2.1 Ipopt::LeastSquareMultipliers::LeastSquareMultipliers (AugSystemSolver & *augSysSolver*)

Constructor.

It needs to be given the strategy object for solving the augmented system.

6.83.2.2 virtual Ipopt::LeastSquareMultipliers::~~LeastSquareMultipliers () [inline], [virtual]

Default destructor.

Definition at line 32 of file `IpLeastSquareMults.hpp`.

6.83.2.3 Ipopt::LeastSquareMultipliers::LeastSquareMultipliers () [private]

Default Constructor.

6.83.2.4 Ipopt::LeastSquareMultipliers::LeastSquareMultipliers (const LeastSquareMultipliers &) [private]

Copy Constructor.

6.83.3 Member Function Documentation

6.83.3.1 virtual bool Ipopt::LeastSquareMultipliers::InitializeImpl (const OptionsList & *options*, const std::string & *prefix*) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::EqMultiplierCalculator](#).

6.83.3.2 virtual bool Ipopt::LeastSquareMultipliers::CalculateMultipliers (Vector & *y_c*, Vector & *y_d*) [virtual]

This method computes the least-square estimates for `y_c` and `y_d` at the current point.

The return value is false, if the least square system could not be solved (the linear system is singular).

Implements [Ipopt::EqMultiplierCalculator](#).

6.83.3.3 void Ipopt::LeastSquareMultipliers::operator= (const LeastSquareMultipliers &) [private]

Overloaded Equals Operator.

6.83.4 Member Data Documentation

6.83.4.1 `SmartPtr<AugSystemSolver> Ipopt::LeastSquareMultipliers::augsysolver_` [private]

Pointer for the augmented system solver to be used for solving the linear system.

Definition at line 68 of file `IpLeastSquareMults.hpp`.

The documentation for this class was generated from the following file:

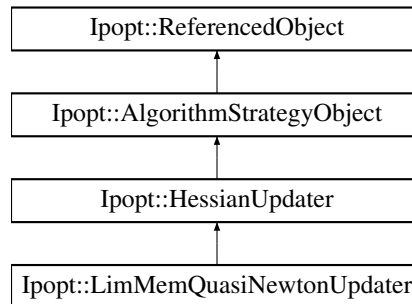
- [Algorithm/IpLeastSquareMults.hpp](#)

6.84 `Ipopt::LimMemQuasiNewtonUpdater` Class Reference

Implementation of the [HessianUpdater](#) for limit-memory quasi-Newton approximation of the Lagrangian Hessian.

`#include <IpLimMemQuasiNewtonUpdater.hpp>`

Inheritance diagram for `Ipopt::LimMemQuasiNewtonUpdater`:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual void [UpdateHessian](#) ()
Update the Hessian based on the current information in IpData.

Constructors/Destructors

- [LimMemQuasiNewtonUpdater](#) (bool update_for_resto)
Default Constructor.
- virtual [~LimMemQuasiNewtonUpdater](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr<RegisteredOptions>](#) roptions)
Methods for [OptionsList](#).

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `LimMemQuasiNewtonUpdater` (const `LimMemQuasiNewtonUpdater` &)
Copy Constructor.
- void `operator=` (const `LimMemQuasiNewtonUpdater` &)
Overloaded Equals Operator.

Auxilliary function

- bool `CheckSkippingBFGS` (`Vector` &s_new, `Vector` &y_new)
Method deciding whether the BFGS update should be skipped.
- bool `UpdateInternalData` (const `Vector` &s_new, const `Vector` &y_new, `SmartPtr`< `Vector` > ypart_new)
Update the internal data, such as the S, Y, L, D etc matrices and vectors that are required for computing the compact representation.
- void `AugmentMultiVector` (`SmartPtr`< `MultiVectorMatrix` > &V, const `Vector` &v_new)
Given a `MultiVector` V, create a new `MultiVectorSpace` with one more column, and return V as a member of that space, consisting of all previous vectors, and in addition v_new in the last column.
- void `AugmentDenseVector` (`SmartPtr`< `DenseVector` > &V, `Number` v_new)
Given a `DenseVector` V, create a new `DenseVectorSpace` with one more row, and return V as a member of that space, consisting of all previous elements, and in addition v_new in the last row.
- void `AugmentLMatrix` (`SmartPtr`< `DenseGenMatrix` > &V, const `MultiVectorMatrix` &S, const `MultiVectorMatrix` &Y)
Given a strictly-lower triangular square `DenseGenMatrix` V, create a new `DenseGenMatrixSpace` with one more dimension, and return V as a member of that space, consisting of all previous elements, and in addition elements $s_i^{\wedge} Ty_j$ for ($i < j$), where s and y are the vectors in the `MultiVectors` S and Y.
- void `AugmentSdotSMatrix` (`SmartPtr`< `DenseSymMatrix` > &V, const `MultiVectorMatrix` &S)
Given a `DenseSymMatrix` V, create a new `DenseGenMatrixSpace` with one more dimension, and return V as a member of that space, consisting of all previous elements, and in addition elements $s_i^{\wedge} Ts_j$ for the new entries, where s are the vectors in the `MultiVector` S.
- void `AugmentSTDRSMMatrix` (`SmartPtr`< `DenseSymMatrix` > &V, const `MultiVectorMatrix` &S, const `MultiVectorMatrix` &DRS)
Given a `DenseSymMatrix` V, create a new `DenseGenMatrixSpace` with one more dimension, and return V as a member of that space, consisting of all previous elements, and in addition elements $s_i^{\wedge} TDRs_j$ for the new entries, where s are the vectors in the `MultiVector` S, and DRs are the vectors in DRS.
- void `ShiftMultiVector` (`SmartPtr`< `MultiVectorMatrix` > &V, const `Vector` &v_new)
Given a `MultiVector` V, get rid of the first column, shift all other columns to the left, and make v_new the last column.
- void `ShiftDenseVector` (`SmartPtr`< `DenseVector` > &V, `Number` v_new)
Given a `DenseVector` V, get rid of the first element, shift all other elements one position to the top, and make v_new the last entry.
- void `ShiftLMatrix` (`SmartPtr`< `DenseGenMatrix` > &V, const `MultiVectorMatrix` &S, const `MultiVectorMatrix` &Y)
Given a strictly-lower triangular square `DenseGenMatrix` V, shift everything one row and column up, and fill the new strictly lower triangular entries as $s_i^{\wedge} Ty_j$ for ($i < j$), where s and y are the vectors in the `MultiVectors` S and Y.
- void `ShiftSdotSMatrix` (`SmartPtr`< `DenseSymMatrix` > &V, const `MultiVectorMatrix` &S)
Given a `DenseSymMatrix` V, shift everything up one row and column, and fill the new entries as $s_i^{\wedge} Ts_j$, where s are the vectors in the `MultiVector` S.
- void `ShiftSTDRSMMatrix` (`SmartPtr`< `DenseSymMatrix` > &V, const `MultiVectorMatrix` &S, const `MultiVectorMatrix` &DRS)
Given a `DenseSymMatrix` V, shift everything up one row and column, and fill the new entries as $s_i^{\wedge} TDRs_j$, where s are the vectors in the `MultiVector` S, and DRs are the vectors in DRS.
- void `RecalcY` (`Number` eta, const `Vector` &DR_x, `MultiVectorMatrix` &S, `MultiVectorMatrix` &Ypart, `SmartPtr`< `MultiVectorMatrix` > &Y)
Method for recomputing Y from scratch, using Ypart (only for restoration phase)
- void `RecalcD` (`MultiVectorMatrix` &S, `MultiVectorMatrix` &Y, `SmartPtr`< `DenseVector` > &D)
Method for recomputing D from S and Y.
- void `RecalcL` (`MultiVectorMatrix` &S, `MultiVectorMatrix` &Y, `SmartPtr`< `DenseGenMatrix` > &L)

- *Method for recomputing L from S and Y.*
- bool [SplitEigenvalues](#) ([DenseGenMatrix](#) &Q, const [DenseVector](#) &E, [SmartPtr](#)< [DenseGenMatrix](#) > &Qminus, [SmartPtr](#)< [DenseGenMatrix](#) > &Qplus)
- *Split the eigenvectors into negative and positive ones.*
- void [StoreInternalDataBackup](#) ()
- *Store a copy of the pointers to the internal data (S, Y, D, L, SdotS, curr_lm_memory) This is called in case the update is started but skipped during the process.*
- void [RestoreInternalDataBackup](#) ()
- *Restore the copy of the pointers to the internal data most recently stored with [StoreInternalDataBackup](#)().*
- void [ReleaseInternalDataBackup](#) ()
- *Release anything that we allocated for [StoreInternalDataBackup](#) and is no longer needed.*
- void [SetW](#) ()
- *Set the W field in [IpData](#) based on the current values of B0_, V_, and U_.*

Private Attributes

- [SmartPtr](#)< const [LowRankUpdateSymMatrixSpace](#) > [h_space_](#)
- *Matrix space for the low-rank Hessian approximation.*
- const bool [update_for_resto_](#)
- *Flag indicating if the update is to be done for the original [NLP](#) or for the restoration phase [NLP](#).*
- [Number](#) [last_eta_](#)
- *Most recent value for eta in the restoration phase objective function (only for [update_for_resto_](#) = true)*
- [SmartPtr](#)< const [Vector](#) > [curr_DR_x_](#)
- *Current DR_x scaling factors in the restoration phase objective function (only for [update_for_resto_](#) = true).*
- [TaggedObject::Tag](#) [curr_DR_x_tag_](#)
- *Tag for [curr_DR_x_](#).*
- [SmartPtr](#)< const [Vector](#) > [curr_red_DR_x_](#)
- *Current DR_x scaling factors in the restoration phase objective function in the smaller space for the approximation - this is only computed if the space is indeed smaller than the x space (only for [update_for_resto_](#) = true)*
- [Number](#) [curr_eta_](#)
- *Current value of weighing factor eta in the restoration phase objective function (only for [update_for_resto_](#) = true)*
- [Index](#) [lm_skipped_iter_](#)
- *Counter for successive iterations in which the update was skipped.*

Information for the limited memory update

- [Index](#) [curr_lm_memory_](#)
- *current size of limited memory*
- [SmartPtr](#)< [MultiVectorMatrix](#) > [S_](#)
- *s pairs for the recent iterations*
- [SmartPtr](#)< [MultiVectorMatrix](#) > [Y_](#)
- *y pairs for the recent iterations.*
- [SmartPtr](#)< [MultiVectorMatrix](#) > [Ypart_](#)
- *For restoration phase update: Y without the quadratic objective function part.*
- [SmartPtr](#)< [DenseVector](#) > [D_](#)
- *Diagonal elements D_k for compact formulation from last update.*
- [SmartPtr](#)< [DenseGenMatrix](#) > [L_](#)
- *Matrix L_k for compact formulation from last update.*
- [SmartPtr](#)< [Vector](#) > [B0_](#)
- *First term (starting matrix) for the approximation.*
- [Number](#) [sigma_](#)

- *First term (starting matrix) for the approximation.*
- [SmartPtr< MultiVectorMatrix > V_](#)
V in LowRankUpdateMatrix from last update.
- [SmartPtr< MultiVectorMatrix > U_](#)
U in LowRankUpdateMatrix from last update.
- [SmartPtr< DenseSymMatrix > SdotS_](#)
For efficient implementation, we store the pairwise products for s's.
- [bool SdotS_uptodate_](#)
Flag indicating whether SdotS_ is update to date from most recent update.
- [SmartPtr< MultiVectorMatrix > DRS_](#)
*DR * S (only for restoration phase)*
- [SmartPtr< DenseSymMatrix > STDRS_](#)
*For efficient implementation, we store the $S^{\wedge} T S DR * S$.*
- [SmartPtr< const Vector > last_x_](#)
Primal variables x from most recent update.
- [SmartPtr< const Vector > last_grad_f_](#)
Gradient of objective function w.r.t.
- [SmartPtr< const Matrix > last_jac_c_](#)
Jacobian for equality constraints w.r.t x at x_last.
- [SmartPtr< const Matrix > last_jac_d_](#)
Jacobian for inequality constraints w.r.t x at x_last.
- [Index curr_lm_memory_old_](#)
current size of limited memory
- [SmartPtr< MultiVectorMatrix > S_old_](#)
s pairs for the recent iterations (backup)
- [SmartPtr< MultiVectorMatrix > Y_old_](#)
y pairs for the recent iterations.
- [SmartPtr< MultiVectorMatrix > Ypart_old_](#)
For restoration phase update: Y without the quadratic objective function part (backup)
- [SmartPtr< DenseVector > D_old_](#)
Diagonal elements D_k for compact formulation from last update (backup).
- [SmartPtr< DenseGenMatrix > L_old_](#)
Matrix L_k for compact formulation from last update (backup).
- [SmartPtr< Vector > B0_old_](#)
First term (starting matrix) for the approximation (backup).
- [Number sigma_old_](#)
First term (starting matrix) for the approximation.
- [SmartPtr< MultiVectorMatrix > V_old_](#)
V in LowRankUpdateMatrix from last update (backup)
- [SmartPtr< MultiVectorMatrix > U_old_](#)
U in LowRankUpdateMatrix from last update (backup)
- [SmartPtr< DenseSymMatrix > SdotS_old_](#)
For efficient implementation, we store the pairwise products for s's (backup).
- [bool SdotS_uptodate_old_](#)
Flag indicating whether SdotS_ is update to date from most recent update (backup).
- [SmartPtr< MultiVectorMatrix > DRS_old_](#)
*DR * S (only for restoration phase) (backup)*
- [SmartPtr< DenseSymMatrix > STDRS_old_](#)
*For efficient implementation, we store the $S^{\wedge} T S DR * S$.*

Algorithmic parameters

- enum `LMUpdateType` { `BFGS` =0, `SR1` }
enumeration for the Hessian update type.
- enum `LMInitialization` {
 `SCALAR1` =0, `SCALAR2`, `SCALAR3`, `SCALAR4`,
 `CONSTANT` }
enumeration for the Hessian initialization.
- `Index limited_memory_max_history_`
Size of memory for limited memory update.
- `LMUpdateType limited_memory_update_type_`
Type of Hessian update.
- `LMInitialization limited_memory_initialization_`
How to choose B0 in the low-rank update.
- `Number limited_memory_init_val_`
Value of B0 (as this multiple of the identity in certain situations.)
- `Index limited_memory_max_skipping_`
Number of successive iterations of skipped updates after which the approximation is reset.
- `Number sigma_safe_min_`
Minimal safeguard value for sigma.
- `Number sigma_safe_max_`
Maximal safeguard value for sigma.
- bool `limited_memory_special_for_resto_`
Flag indicating if Hessian approximation should be done in a special manner for the restoration phase.

Additional Inherited Members

6.84.1 Detailed Description

Implementation of the `HessianUpdater` for limit-memory quasi-Newton approximation of the Lagrangian Hessian.
Definition at line 25 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.2 Member Enumeration Documentation

6.84.2.1 enum `Ipopt::LimMemQuasiNewtonUpdater::LMUpdateType` `[private]`

enumeration for the Hessian update type.

Enumerator

BFGS

SR1

Definition at line 75 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.2.2 enum Ipopt::LimMemQuasiNewtonUpdater::LMInitialization [private]

enumeration for the Hessian initialization.

Enumerator

SCALAR1

SCALAR2

SCALAR3

SCALAR4

CONSTANT

Definition at line 83 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.3 Constructor & Destructor Documentation

6.84.3.1 Ipopt::LimMemQuasiNewtonUpdater::LimMemQuasiNewtonUpdater (bool *update_for_resto*)

Default Constructor.

6.84.3.2 virtual Ipopt::LimMemQuasiNewtonUpdater::~~LimMemQuasiNewtonUpdater () [inline],[virtual]

Default destructor.

Definition at line 34 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.3.3 Ipopt::LimMemQuasiNewtonUpdater::LimMemQuasiNewtonUpdater (const LimMemQuasiNewtonUpdater &) [private]

Copy Constructor.

6.84.4 Member Function Documentation

6.84.4.1 virtual bool Ipopt::LimMemQuasiNewtonUpdater::Initializeml (const OptionsList & *options*, const std::string & *prefix*) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::HessianUpdater](#).

6.84.4.2 virtual void Ipopt::LimMemQuasiNewtonUpdater::UpdateHessian () [virtual]

Update the Hessian based on the current information in IpData.

Implements [Ipopt::HessianUpdater](#).

6.84.4.3 static void Ipopt::LimMemQuasiNewtonUpdater::RegisterOptions (SmartPtr< RegisteredOptions > *roptions*) [static]

Methods for [OptionsList](#).

6.84.4.4 void Ipopt::LimMemQuasiNewtonUpdater::operator= (const LimMemQuasiNewtonUpdater &) [private]

Overloaded Equals Operator.

6.84.4.5 `bool Ipopt::LimMemQuasiNewtonUpdater::CheckSkippingBFGS (Vector & s_new, Vector & y_new) [private]`

Method deciding whether the BFGS update should be skipped.

It returns true, if no update is to be performed this time. If Powell-damping is performed, the Vectors `s_new` and `y_new`, might be adapted.

6.84.4.6 `bool Ipopt::LimMemQuasiNewtonUpdater::UpdateInternalData (const Vector & s_new, const Vector & y_new, SmartPtr< Vector > ypart_new) [private]`

Update the internal data, such as the S, Y, L, D etc matrices and vectors that are required for computing the compact representation.

The method returns true if the limited memory history grew (i.e., `curr_lm_memory_` was increased).

6.84.4.7 `void Ipopt::LimMemQuasiNewtonUpdater::AugmentMultiVector (SmartPtr< MultiVectorMatrix > & V, const Vector & v_new) [private]`

Given a MultiVector V, create a new MultiVectorSpace with one more column, and return V as a member of that space, consisting of all previous vectors, and in addition `v_new` in the last column.

If V is NULL, then a new [MatrixSpace](#) with one column is created.

6.84.4.8 `void Ipopt::LimMemQuasiNewtonUpdater::AugmentDenseVector (SmartPtr< DenseVector > & V, Number v_new) [private]`

Given a [DenseVector](#) V, create a new [DenseVectorSpace](#) with one more row, and return V as a member of that space, consisting of all previous elements, and in addition `v_new` in the last row.

If V is NULL, then a new [DenseVectorSpace](#) with dimension one is created.

6.84.4.9 `void Ipopt::LimMemQuasiNewtonUpdater::AugmentLMatrix (SmartPtr< DenseGenMatrix > & V, const MultiVectorMatrix & S, const MultiVectorMatrix & Y) [private]`

Given a strictly-lower triangular square [DenseGenMatrix](#) V, create a new [DenseGenMatrixSpace](#) with one more dimension, and return V as a member of that space, consisting of all previous elements, and in addition elements $s_i^T y_j$ for $(i < j)$, where s and y are the vectors in the MultiVectors S and Y.

If V is NULL, then a new [DenseGenMatrixSpace](#) with dimension one is created.

6.84.4.10 `void Ipopt::LimMemQuasiNewtonUpdater::AugmentSdotSMatrix (SmartPtr< DenseSymMatrix > & V, const MultiVectorMatrix & S) [private]`

Given a [DenseSymMatrix](#) V, create a new [DenseGenMatrixSpace](#) with one more dimension, and return V as a member of that space, consisting of all previous elements, and in addition elements $s_i^T s_j$ for the new entries, where s are the vectors in the MultiVector S.

If V is NULL, then a new [DenseGenMatrixSpace](#) with dimension one is created.

6.84.4.11 `void Ipopt::LimMemQuasiNewtonUpdater::AugmentSTDRSMMatrix (SmartPtr< DenseSymMatrix > & V, const MultiVectorMatrix & S, const MultiVectorMatrix & DRS) [private]`

Given a [DenseSymMatrix](#) V, create a new [DenseGenMatrixSpace](#) with one more dimension, and return V as a member of that space, consisting of all previous elements, and in addition elements $s_i^T DRS_j$ for the new entries, where s are the vectors in the MultiVector S, and DRs are the vectors in DRS.

If V is NULL, then a new [DenseGenMatrixSpace](#) with dimension one is created.

6.84.4.12 `void Ipopt::LimMemQuasiNewtonUpdater::ShiftMultiVector (SmartPtr< MultiVectorMatrix > & V, const Vector & v_new) [private]`

Given a MultiVector V, get rid of the first column, shift all other columns to the left, and make v_new the last column. The entity that V points to at the call, is not changed - a new entity is created in the method and returned as V.

6.84.4.13 `void Ipopt::LimMemQuasiNewtonUpdater::ShiftDenseVector (SmartPtr< DenseVector > & V, Number v_new) [private]`

Given a DenseVector V, get rid of the first element, shift all other elements one position to the top, and make v_new the last entry.

The entity that V points to at the call, is not changed - a new entity is created in the method and returned as V.

6.84.4.14 `void Ipopt::LimMemQuasiNewtonUpdater::ShiftLMatrix (SmartPtr< DenseGenMatrix > & V, const MultiVectorMatrix & S, const MultiVectorMatrix & Y) [private]`

Given a strictly-lower triangular square DenseGenMatrix V, shift everything one row and column up, and fill the new strictly lower triangular entries as $s_i^{Ty_j}$ for $(i < j)$, where s and y are the vectors in the MultiVectors S and Y.

The entity that V points to at the call, is not changed - a new entity is created in the method and returned as V.

6.84.4.15 `void Ipopt::LimMemQuasiNewtonUpdater::ShiftSdotSMatrix (SmartPtr< DenseSymMatrix > & V, const MultiVectorMatrix & S) [private]`

Given a DenseSymMatrix V, shift everything up one row and column, and fill the new entries as $s_i^{Ts_j}$, where s are the vectors in the MultiVector S.

The entity that V points to at the call, is not changed - a new entity is created in the method and returned as V.

6.84.4.16 `void Ipopt::LimMemQuasiNewtonUpdater::ShiftSTDRSMatrix (SmartPtr< DenseSymMatrix > & V, const MultiVectorMatrix & S, const MultiVectorMatrix & DRS) [private]`

Given a DenseSymMatrix V, shift everything up one row and column, and fill the new entries as $s_i^{TDRs_j}$, where s are the vectors in the MultiVector S, and DRs are the vectors in DRS.

The entity that V points to at the call, is not changed - a new entity is created in the method and returned as V.

6.84.4.17 `void Ipopt::LimMemQuasiNewtonUpdater::RecalcY (Number eta, const Vector & DR_x, MultiVectorMatrix & S, MultiVectorMatrix & Ypart, SmartPtr< MultiVectorMatrix > & Y) [private]`

Method for recomputing Y from scratch, using Ypart (only for restoration phase)

6.84.4.18 `void Ipopt::LimMemQuasiNewtonUpdater::RecalcD (MultiVectorMatrix & S, MultiVectorMatrix & Y, SmartPtr< DenseVector > & D) [private]`

Method for recomputing D from S and Y.

6.84.4.19 `void Ipopt::LimMemQuasiNewtonUpdater::RecalcL (MultiVectorMatrix & S, MultiVectorMatrix & Y, SmartPtr< DenseGenMatrix > & L) [private]`

Method for recomputing L from S and Y.

6.84.4.20 `bool Ipopt::LimMemQuasiNewtonUpdater::SplitEigenvalues (DenseGenMatrix & Q, const DenseVector & E, SmartPtr< DenseGenMatrix > & Qminus, SmartPtr< DenseGenMatrix > & Qplus) [private]`

Split the eigenvectors into negative and positive ones.

Given the eigenvectors in Q and the eigenvalues (in ascending order) in, this returns Q_{minus} as the negative eigenvectors times $\sqrt{-\text{eval}}$, and Q_{plus} as the positive eigenvectors times $\sqrt{\text{eval}}$. If Q_{minus} or Q_{plus} is NULL, it means that there are not negative or positive eigenvalues. Q might be changed during this call. The return value is true, if the ratio of the smallest over the largest eigenvalue (in absolute values) is too small; in that case, the update should be skipped.

6.84.4.21 `void Ipopt::LimMemQuasiNewtonUpdater::StoreInternalDataBackup () [private]`

Store a copy of the pointers to the internal data (S , Y , D , L , $S_{\text{dot}}S$, curr_lm_memory) This is called in case the update is started but skipped during the process.

6.84.4.22 `void Ipopt::LimMemQuasiNewtonUpdater::RestoreInternalDataBackup () [private]`

Restore the copy of the pointers to the internal data most recently stored with [StoreInternalDataBackup\(\)](#).

6.84.4.23 `void Ipopt::LimMemQuasiNewtonUpdater::ReleaseInternalDataBackup () [private]`

Release anything that we allocated for `StoreInternalDataBackup` and is no longer needed.

6.84.4.24 `void Ipopt::LimMemQuasiNewtonUpdater::SetW () [private]`

Set the W field in `IpData` based on the current values of $B0$, V , and U .

6.84.5 Member Data Documentation

6.84.5.1 `SmartPtr<const LowRankUpdateSymMatrixSpace> Ipopt::LimMemQuasiNewtonUpdater::h_space_ [private]`

[Matrix](#) space for the low-rank Hessian approximation.

Definition at line 68 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.2 `Index Ipopt::LimMemQuasiNewtonUpdater::limited_memory_max_history_ [private]`

Size of memory for limited memory update.

Definition at line 73 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.3 `LMUpdateType Ipopt::LimMemQuasiNewtonUpdater::limited_memory_update_type_ [private]`

Type of Hessian update.

Definition at line 81 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.4 `LMInitialization Ipopt::LimMemQuasiNewtonUpdater::limited_memory_initialization_ [private]`

How to choose $B0$ in the low-rank update.

Definition at line 92 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.5 `Number Ipopt::LimMemQuasiNewtonUpdater::limited_memory_init_val_ [private]`

Value of $B0$ (as this multiple of the identity in certain situations.)

Definition at line 95 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.6 `Index Ipopt::LimMemQuasiNewtonUpdater::limited_memory_max_skipping_ [private]`

Number of successive iterations of skipped updates after which the approximation is reset.

Definition at line 98 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.7 **Number** Ipopt::LimMemQuasiNewtonUpdater::sigma_safe_min_ [private]

Minimal safeguard value for sigma.

Definition at line 100 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.8 **Number** Ipopt::LimMemQuasiNewtonUpdater::sigma_safe_max_ [private]

Maximal safeguard value for sigma.

Definition at line 102 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.9 **bool** Ipopt::LimMemQuasiNewtonUpdater::limited_memory_special_for_resto_ [private]

Flag indicating if Hessian approximation should be done in a special manner for the restoration phase.

Definition at line 105 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.10 **const bool** Ipopt::LimMemQuasiNewtonUpdater::update_for_resto_ [private]

Flag indicating if the update is to be done for the original [NLP](#) or for the restoration phase [NLP](#).

In the latter case, we are performing a "structured" update, taking into account the first explicit term in the objective function of the form $\eta * D_r * x_k$

Definition at line 113 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.11 **Number** Ipopt::LimMemQuasiNewtonUpdater::last_eta_ [private]

Most recent value for eta in the restoration phase objective function (only for update_for_resto_ = true)

Definition at line 116 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.12 **SmartPtr<const Vector>** Ipopt::LimMemQuasiNewtonUpdater::curr_DR_x_ [private]

Current DR_x scaling factors in the restoration phase objective function (only for update_for_resto_ = true).

This should not change throughout one restoration phase.

Definition at line 120 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.13 **TaggedObject::Tag** Ipopt::LimMemQuasiNewtonUpdater::curr_DR_x_tag_ [private]

Tag for curr_DR_x_.

Definition at line 122 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.14 **SmartPtr<const Vector>** Ipopt::LimMemQuasiNewtonUpdater::curr_red_DR_x_ [private]

Current DR_x scaling factors in the restoration phase objective function in the smaller space for the approximation - this is only computed if the space is indeed smaller than the x space (only for update_for_resto_ = true)

Definition at line 127 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.15 **Number** Ipopt::LimMemQuasiNewtonUpdater::curr_eta_ [private]

Current value of weighing factor eta in the restoration phase objective function (only for update_for_resto_ = true)

Definition at line 130 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.16 Index `Ipopt::LimMemQuasiNewtonUpdater::lm_skipped_iter_` [private]

Counter for successive iterations in which the update was skipped.

Definition at line 134 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.17 Index `Ipopt::LimMemQuasiNewtonUpdater::curr_lm_memory_` [private]

current size of limited memory

Definition at line 139 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.18 SmartPtr<MultiVectorMatrix> `Ipopt::LimMemQuasiNewtonUpdater::S_` [private]

s pairs for the recent iterations

Definition at line 141 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.19 SmartPtr<MultiVectorMatrix> `Ipopt::LimMemQuasiNewtonUpdater::Y_` [private]

y pairs for the recent iterations.

If `update_for_resto` is true, then this includes only the information for the constraints.

Definition at line 145 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.20 SmartPtr<MultiVectorMatrix> `Ipopt::LimMemQuasiNewtonUpdater::Ypart_` [private]

For restoration phase update: Y without the quadratic objective function part.

Definition at line 148 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.21 SmartPtr<DenseVector> `Ipopt::LimMemQuasiNewtonUpdater::D_` [private]

Diagonal elements D_k for compact formulation from last update.

Definition at line 151 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.22 SmartPtr<DenseGenMatrix> `Ipopt::LimMemQuasiNewtonUpdater::L_` [private]

[Matrix](#) L_k for compact formulation from last update.

Definition at line 153 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.23 SmartPtr<Vector> `Ipopt::LimMemQuasiNewtonUpdater::B0_` [private]

First term (starting matrix) for the approximation.

Definition at line 155 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.24 Number `Ipopt::LimMemQuasiNewtonUpdater::sigma_` [private]

First term (starting matrix) for the approximation.

If that first terms is a multiple of the identity, sigma give that factor. Otherwise sigma = -1.

Definition at line 159 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.25 SmartPtr<MultiVectorMatrix> `Ipopt::LimMemQuasiNewtonUpdater::V_` [private]

V in `LowRankUpdateMatrix` from last update.

Definition at line 161 of file `IpLimMemQuasiNewtonUpdater.hpp`.

6.84.5.26 SmartPtr<MultiVectorMatrix> Ipopt::LimMemQuasiNewtonUpdater::U_ [private]

U in LowRankUpdateMatrix from last update.

Definition at line 163 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.27 SmartPtr<DenseSymMatrix> Ipopt::LimMemQuasiNewtonUpdater::SdotS_ [private]

For efficient implementation, we store the pairwise products for s's.

Definition at line 166 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.28 bool Ipopt::LimMemQuasiNewtonUpdater::SdotS_uptodate_ [private]

Flag indicating whether SdotS_ is update to date from most recent update.

Definition at line 169 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.29 SmartPtr<MultiVectorMatrix> Ipopt::LimMemQuasiNewtonUpdater::DRS_ [private]

$DR * S$ (only for restoration phase)

Definition at line 171 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.30 SmartPtr<DenseSymMatrix> Ipopt::LimMemQuasiNewtonUpdater::STDRS_ [private]

For efficient implementation, we store the $S^T S DR * S$.

Only for restoration phase.

Definition at line 174 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.31 SmartPtr<const Vector> Ipopt::LimMemQuasiNewtonUpdater::last_x_ [private]

Primal variables x from most recent update.

Definition at line 176 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.32 SmartPtr<const Vector> Ipopt::LimMemQuasiNewtonUpdater::last_grad_f_ [private]

Gradient of objective function w.r.t.

x at x_last_

Definition at line 178 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.33 SmartPtr<const Matrix> Ipopt::LimMemQuasiNewtonUpdater::last_jac_c_ [private]

Jacobian for equality constraints w.r.t x at x_last.

Definition at line 180 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.34 SmartPtr<const Matrix> Ipopt::LimMemQuasiNewtonUpdater::last_jac_d_ [private]

Jacobian for inequality constraints w.r.t x at x_last.

Definition at line 182 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.35 Index Ipopt::LimMemQuasiNewtonUpdater::curr_lm_memory_old_ [private]

current size of limited memory

Definition at line 184 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.36 SmartPtr<MultiVectorMatrix> Ipopt::LimMemQuasiNewtonUpdater::S_old_ [private]

s pairs for the recent iterations (backup)

Definition at line 186 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.37 SmartPtr<MultiVectorMatrix> Ipopt::LimMemQuasiNewtonUpdater::Y_old_ [private]

y pairs for the recent iterations.

If update_for_resto is true, then this includes only the information for the constraints. (backup)

Definition at line 190 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.38 SmartPtr<MultiVectorMatrix> Ipopt::LimMemQuasiNewtonUpdater::Ypart_old_ [private]

For restoration phase update: Y without the quadratic objective function part (backup)

Definition at line 193 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.39 SmartPtr<DenseVector> Ipopt::LimMemQuasiNewtonUpdater::D_old_ [private]

Diagonal elements D_k for compact formulation from last update (backup).

Definition at line 196 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.40 SmartPtr<DenseGenMatrix> Ipopt::LimMemQuasiNewtonUpdater::L_old_ [private]

[Matrix](#) L_k for compact formulation from last update (backup).

Definition at line 198 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.41 SmartPtr<Vector> Ipopt::LimMemQuasiNewtonUpdater::B0_old_ [private]

First term (starting matrix) for the approximation (backup).

Definition at line 200 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.42 Number Ipopt::LimMemQuasiNewtonUpdater::sigma_old_ [private]

First term (starting matrix) for the approximation.

If that first terms is a multiple of the identity, sigma give that factor. Otherwise sigma = -1. (backup)

Definition at line 204 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.43 SmartPtr<MultiVectorMatrix> Ipopt::LimMemQuasiNewtonUpdater::V_old_ [private]

V in LowRankUpdateMatrix from last update (backup)

Definition at line 206 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.44 SmartPtr<MultiVectorMatrix> Ipopt::LimMemQuasiNewtonUpdater::U_old_ [private]

U in LowRankUpdateMatrix from last update (backup)

Definition at line 208 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.45 SmartPtr<DenseSymMatrix> Ipopt::LimMemQuasiNewtonUpdater::SdotS_old_ [private]

For efficient implementation, we store the pairwise products for s's (backup).

Definition at line 211 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.46 `bool Ipopt::LimMemQuasiNewtonUpdater::SdotS_uptodate_old_ [private]`

Flag indicating whether SdotS_ is update to date from most recent update (backup).

Definition at line 214 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.47 `SmartPtr<MultiVectorMatrix> Ipopt::LimMemQuasiNewtonUpdater::DRS_old_ [private]`

$DR * S$ (only for restoration phase) (backup)

Definition at line 216 of file IpLimMemQuasiNewtonUpdater.hpp.

6.84.5.48 `SmartPtr<DenseSymMatrix> Ipopt::LimMemQuasiNewtonUpdater::STDRS_old_ [private]`

For efficient implementation, we store the $S^T S DR * S$.

Only for restoration phase. (backup)

Definition at line 219 of file IpLimMemQuasiNewtonUpdater.hpp.

The documentation for this class was generated from the following file:

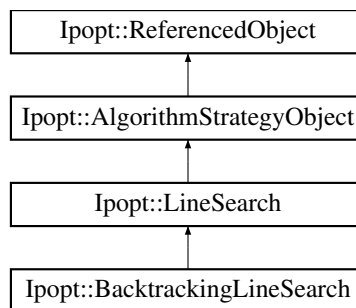
- [Algorithm/IpLimMemQuasiNewtonUpdater.hpp](#)

6.85 Ipopt::LineSearch Class Reference

Base class for line search objects.

```
#include <IpLineSearch.hpp>
```

Inheritance diagram for Ipopt::LineSearch:



Public Member Functions

- virtual void [FindAcceptableTrialPoint](#) ()=0
Perform the line search.
- virtual void [Reset](#) ()=0
Reset the line search.
- virtual void [SetRigorousLineSearch](#) (bool rigorous)=0
Set flag indicating whether a very rigorous line search should be performed.
- virtual bool [CheckSkippedLineSearch](#) ()=0
Check if the line search procedure didn't accept a new iterate during the last call of [FindAcceptableTrialPoint](#)().
- virtual bool [ActivateFallbackMechanism](#) ()=0

This method should be called if the optimization process requires the line search object to switch to some fallback mechanism (like the restoration phase), when the regular optimization procedure cannot be continued (for example, because the search direction could not be computed).

Constructors/Destructors

- [LineSearch](#) ()
Default Constructor.
- virtual [~LineSearch](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [LineSearch](#) (const [LineSearch](#) &)
Copy Constructor.
- void [operator=](#) (const [LineSearch](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.85.1 Detailed Description

Base class for line search objects.

Definition at line 20 of file IpLineSearch.hpp.

6.85.2 Constructor & Destructor Documentation

6.85.2.1 Ipopt::LineSearch::LineSearch () [inline]

Default Constructor.

Definition at line 26 of file IpLineSearch.hpp.

6.85.2.2 virtual Ipopt::LineSearch::~~LineSearch () [inline],[virtual]

Default destructor.

Definition at line 30 of file IpLineSearch.hpp.

6.85.2.3 Ipopt::LineSearch::LineSearch (const LineSearch &) [private]

Copy Constructor.

6.85.3 Member Function Documentation

6.85.3.1 virtual void Ipopt::LineSearch::FindAcceptableTrialPoint () [pure virtual]

Perform the line search.

As search direction the delta in the data object is used

Implemented in [Ipopt::BacktrackingLineSearch](#).

6.85.3.2 `virtual void Ipopt::LineSearch::Reset () [pure virtual]`

Reset the line search.

This function should be called if all previous information should be discarded when the line search is performed the next time. For example, this method should be called after the barrier parameter is changed.

Implemented in [Ipopt::BacktrackingLineSearch](#).

6.85.3.3 `virtual void Ipopt::LineSearch::SetRigorousLineSearch (bool rigorous) [pure virtual]`

Set flag indicating whether a very rigorous line search should be performed.

If this flag is set to true, the line search algorithm might decide to abort the line search and not to accept a new iterate. If the line search decided not to accept a new iterate, the return value of [CheckSkippedLineSearch\(\)](#) is true at the next call. For example, in the non-monotone barrier parameter update procedure, the filter algorithm should not switch to the restoration phase in the free mode; instead, the algorithm should switch to the fixed mode.

Implemented in [Ipopt::BacktrackingLineSearch](#).

6.85.3.4 `virtual bool Ipopt::LineSearch::CheckSkippedLineSearch () [pure virtual]`

Check if the line search procedure didn't accept a new iterate during the last call of [FindAcceptableTrialPoint\(\)](#).

Implemented in [Ipopt::BacktrackingLineSearch](#).

6.85.3.5 `virtual bool Ipopt::LineSearch::ActivateFallbackMechanism () [pure virtual]`

This method should be called if the optimization process requires the line search object to switch to some fallback mechanism (like the restoration phase), when the regular optimization procedure cannot be continued (for example, because the search direction could not be computed).

This will cause the line search object to immediately proceed with this mechanism when [FindAcceptableTrialPoint\(\)](#) is call. This method returns false if no fallback mechanism is available.

Implemented in [Ipopt::BacktrackingLineSearch](#).

6.85.3.6 `void Ipopt::LineSearch::operator= (const LineSearch &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

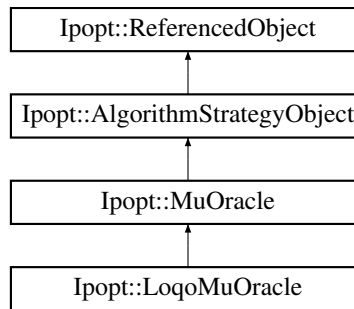
- [Algorithm/IpLineSearch.hpp](#)

6.86 Ipopt::LoqoMuOracle Class Reference

Implementation of the LOQO formula for computing the barrier parameter.

```
#include <IpLoqoMuOracle.hpp>
```

Inheritance diagram for Ipopt::LoqoMuOracle:



Public Member Functions

- virtual bool **InitializeImpl** (const [OptionsList](#) &options, const std::string &prefix)
Initialize method - overloaded from [AlgorithmStrategyObject](#).
- virtual bool **CalculateMu** ([Number](#) mu_min, [Number](#) mu_max, [Number](#) &new_mu)
Method for computing the value of the barrier parameter that could be used in the current iteration (using the LOQO formula).

Constructors/Destructors

- [LoqoMuOracle](#) ()
Default Constructor.
- virtual [~LoqoMuOracle](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [LoqoMuOracle](#) (const [LoqoMuOracle](#) &)
Copy Constructor.
- void **operator=** (const [LoqoMuOracle](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.86.1 Detailed Description

Implementation of the LOQO formula for computing the barrier parameter.

Definition at line 20 of file IpLoqoMuOracle.hpp.

6.86.2 Constructor & Destructor Documentation

6.86.2.1 Ipopt::LoqoMuOracle::LoqoMuOracle ()

Default Constructor.

6.86.2.2 `virtual Ipopt::LoqoMuOracle::~~LoqoMuOracle () [virtual]`

Default destructor.

6.86.2.3 `Ipopt::LoqoMuOracle::LoqoMuOracle (const LoqoMuOracle &) [private]`

Copy Constructor.

6.86.3 Member Function Documentation

6.86.3.1 `virtual bool Ipopt::LoqoMuOracle::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

Initialize method - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::MuOracle](#).

6.86.3.2 `virtual bool Ipopt::LoqoMuOracle::CalculateMu (Number mu_min, Number mu_max, Number & new_mu) [virtual]`

Method for computing the value of the barrier parameter that could be used in the current iteration (using the LOQO formula).

Implements [Ipopt::MuOracle](#).

6.86.3.3 `void Ipopt::LoqoMuOracle::operator= (const LoqoMuOracle &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

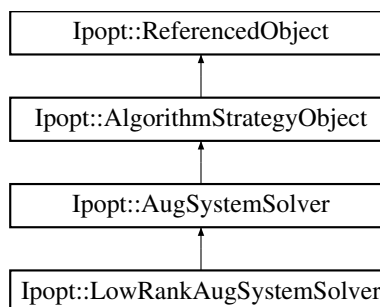
- [Algorithm/IpLoqoMuOracle.hpp](#)

6.87 Ipopt::LowRankAugSystemSolver Class Reference

Solver for the augmented system with [LowRankUpdateSymMatrix](#) Hessian matrices.

```
#include <IpLowRankAugSystemSolver.hpp>
```

Inheritance diagram for Ipopt::LowRankAugSystemSolver:



Public Member Functions

- `bool` [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)

overloaded from [AlgorithmStrategyObject](#)

- virtual [ESymSolverStatus Solve](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) *J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) *J_d, const [Vector](#) *D_d, double delta_d, const [Vector](#) &rhs_x, const [Vector](#) &rhs_s, const [Vector](#) &rhs_c, const [Vector](#) &rhs_d, [Vector](#) &sol_x, [Vector](#) &sol_s, [Vector](#) &sol_c, [Vector](#) &sol_d, bool check_NegEVals, [Index](#) numberOfNegEVals)

Set up the augmented system and solve it for a given right hand side.

- virtual [Index NumberOfNegEVals](#) () const

Number of negative eigenvalues detected during last solve.

- virtual bool [ProvidesInertia](#) () const

Query whether inertia is computed by linear solver.

- virtual bool [IncreaseQuality](#) ()

Request to increase quality of solution for next solve.

Constructors/Destructors

- [LowRankAugSystemSolver](#) ([AugSystemSolver](#) &aug_system_solver)

Constructor using only a linear solver object.

- virtual [~LowRankAugSystemSolver](#) ()

Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [LowRankAugSystemSolver](#) ()
Default constructor.
- [LowRankAugSystemSolver](#) (const [LowRankAugSystemSolver](#) &)
Copy Constructor.
- void [operator=](#) (const [LowRankAugSystemSolver](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus UpdateFactorization](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d, const [Vector](#) &proto_rhs_x, const [Vector](#) &proto_rhs_s, const [Vector](#) &proto_rhs_c, const [Vector](#) &proto_rhs_d, bool check_NegEVals, [Index](#) numberOfNegEVals)

Method for updating the factorization, including J1_, J2_, Vtilde1_, Utilde2, Wdiag_, compound_sol_vecspace_.

- [ESymSolverStatus SolveMultiVector](#) (const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d, const [Vector](#) &proto_rhs_x, const [Vector](#) &proto_rhs_s, const [Vector](#) &proto_rhs_c, const [Vector](#) &proto_rhs_d, const [MultiVectorMatrix](#) &V, const [SmartPtr](#)< const [Matrix](#) > &P_LM, [SmartPtr](#)< [MultiVectorMatrix](#) > &V_x, [SmartPtr](#)< [MultiVectorMatrix](#) > &Vtilde1, [SmartPtr](#)< [MultiVectorMatrix](#) > &Vtilde1_x, bool check_NegEVals, [Index](#) numberOfNegEVals)

Method for solving the augmented system without low-rank update for multiple right hand sides that are provided as [MultiVectorMatrix](#).

- bool [AugmentedSystemRequiresChange](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d)

Method that compares the tags of the data for the matrix with those from the previous call.

Private Attributes

- [SmartPtr< AugSystemSolver > aug_system_solver_](#)
The augmented system solver object that should be used for the factorization of the augmented system without the low-rank update.
- [Index num_neg_evals_](#)
Stores the number of negative eigenvalues detected during most recent factorization.

Tags and values to track in order to decide whether the

matrix has to be updated compared to the most recent call of the Set method.

- [TaggedObject::Tag w_tag_](#)
Tag for W matrix.
- double [w_factor_](#)
Most recent value of W_factor.
- [TaggedObject::Tag d_x_tag_](#)
Tag for D_x vector, representing the diagonal matrix D_x.
- double [delta_x_](#)
Most recent value of delta_x from Set method.
- [TaggedObject::Tag d_s_tag_](#)
Tag for D_s vector, representing the diagonal matrix D_s.
- double [delta_s_](#)
Most recent value of delta_s from Set method.
- [TaggedObject::Tag j_c_tag_](#)
Tag for J_c matrix.
- [TaggedObject::Tag d_c_tag_](#)
Tag for D_c vector, representing the diagonal matrix D_c.
- double [delta_c_](#)
Most recent value of delta_c from Set method.
- [TaggedObject::Tag j_d_tag_](#)
Tag for J_d matrix.
- [TaggedObject::Tag d_d_tag_](#)
Tag for D_d vector, representing the diagonal matrix D_d.
- double [delta_d_](#)
Most recent value of delta_d from Set method.

Information to be stored in order to resolve for the

same matrix with a different right hand side.

- bool [first_call_](#)
- [SmartPtr< DenseGenMatrix > J1_](#)
- [SmartPtr< DenseGenMatrix > J2_](#)
- [SmartPtr< MultiVectorMatrix > Vtilde1_](#)
- [SmartPtr< MultiVectorMatrix > Utilde2_](#)
- [SmartPtr< DiagMatrix > Wdiag_](#)
Hessian Matrix passed to the augmented system solver solving the matrix without the low-rank update.
- [SmartPtr< const CompoundVectorSpace > compound_sol_vecspace_](#)
Vector space for Compound vectors that capture the entire right hand side and solution vectors .

Additional Inherited Members

6.87.1 Detailed Description

Solver for the augmented system with [LowRankUpdateSymMatrix](#) Hessian matrices.

This version works with the Sherman-Morrison formula and multiple backsolves.

Definition at line 24 of file `IpLowRankAugSystemSolver.hpp`.

6.87.2 Constructor & Destructor Documentation

6.87.2.1 `Ipopt::LowRankAugSystemSolver::LowRankAugSystemSolver (AugSystemSolver & aug_system_solver)`

Constructor using only a linear solver object.

6.87.2.2 `virtual Ipopt::LowRankAugSystemSolver::~~LowRankAugSystemSolver () [virtual]`

Default destructor.

6.87.2.3 `Ipopt::LowRankAugSystemSolver::LowRankAugSystemSolver () [private]`

Default constructor.

6.87.2.4 `Ipopt::LowRankAugSystemSolver::LowRankAugSystemSolver (const LowRankAugSystemSolver &) [private]`

Copy Constructor.

6.87.3 Member Function Documentation

6.87.3.1 `bool Ipopt::LowRankAugSystemSolver::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AugSystemSolver](#).

6.87.3.2 `virtual ESymSolverStatus Ipopt::LowRankAugSystemSolver::Solve (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix * J_c, const Vector * D_c, double delta_c, const Matrix * J_d, const Vector * D_d, double delta_d, const Vector & rhs_x, const Vector & rhs_s, const Vector & rhs_c, const Vector & rhs_d, Vector & sol_x, Vector & sol_s, Vector & sol_c, Vector & sol_d, bool check_NegEvals, Index numberOfNegEvals) [virtual]`

Set up the augmented system and solve it for a given right hand side.

Reimplemented from [Ipopt::AugSystemSolver](#).

6.87.3.3 `virtual Index Ipopt::LowRankAugSystemSolver::NumberOfNegEvals () const [virtual]`

Number of negative eigenvalues detected during last solve.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see `ProvidesInertia`).

Implements [Ipopt::AugSystemSolver](#).

6.87.3.4 `virtual bool Ipopt::LowRankAugSystemSolver::ProvidesInertia () const [virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::AugSystemSolver](#).

6.87.3.5 `virtual bool Ipopt::LowRankAugSystemSolver::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

Ask underlying linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::AugSystemSolver](#).

6.87.3.6 `void Ipopt::LowRankAugSystemSolver::operator= (const LowRankAugSystemSolver &) [private]`

Overloaded Equals Operator.

6.87.3.7 `ESymSolverStatus Ipopt::LowRankAugSystemSolver::UpdateFactorization (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix & J_c, const Vector * D_c, double delta_c, const Matrix & J_d, const Vector * D_d, double delta_d, const Vector & proto_rhs_x, const Vector & proto_rhs_s, const Vector & proto_rhs_c, const Vector & proto_rhs_d, bool check_NegEVals, Index numberOfNegEVals) [private]`

Method for updating the factorization, including J1_, J2_, Vtilde1_, Utilde2, Wdiag_, compound_sol_vecspace_.

6.87.3.8 `ESymSolverStatus Ipopt::LowRankAugSystemSolver::SolveMultiVector (const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix & J_c, const Vector * D_c, double delta_c, const Matrix & J_d, const Vector * D_d, double delta_d, const Vector & proto_rhs_x, const Vector & proto_rhs_s, const Vector & proto_rhs_c, const Vector & proto_rhs_d, const MultiVectorMatrix & V, const SmartPtr< const Matrix > & P_LM, SmartPtr< MultiVectorMatrix > & V_x, SmartPtr< MultiVectorMatrix > & Vtilde1, SmartPtr< MultiVectorMatrix > & Vtilde1_x, bool check_NegEVals, Index numberOfNegEVals) [private]`

Method for solving the augmented system without low-rank update for multiple right hand sides that are provided as [MultiVectorMatrix](#).

The result is returned as a [MultiVectorMatrix](#) in Vtilde1. V_x and Vtilde1_x are V and Vtilde1 in the x-space.

6.87.3.9 `bool Ipopt::LowRankAugSystemSolver::AugmentedSystemRequiresChange (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix & J_c, const Vector * D_c, double delta_c, const Matrix & J_d, const Vector * D_d, double delta_d) [private]`

Method that compares the tags of the data for the matrix with those from the previous call.

Returns true, if there was a change and the factorization has to be updated.

6.87.4 Member Data Documentation

6.87.4.1 `SmartPtr<AugSystemSolver> Ipopt::LowRankAugSystemSolver::aug_system_solver_ [private]`

The augmented system solver object that should be used for the factorization of the augmented system without the low-rank update.

Definition at line 110 of file IpLowRankAugSystemSolver.hpp.

6.87.4.2 TaggedObject::Tag Ipopt::LowRankAugSystemSolver::w_tag_ [private]

Tag for W matrix.

If W has been given to Set as NULL, then this tag is set to 0

Definition at line 120 of file IpLowRankAugSystemSolver.hpp.

6.87.4.3 double Ipopt::LowRankAugSystemSolver::w_factor_ [private]

Most recent value of W_factor.

Definition at line 122 of file IpLowRankAugSystemSolver.hpp.

6.87.4.4 TaggedObject::Tag Ipopt::LowRankAugSystemSolver::d_x_tag_ [private]

Tag for D_x vector, representing the diagonal matrix D_x.

If D_x has been given to Set as NULL, then this tag is set to 0

Definition at line 126 of file IpLowRankAugSystemSolver.hpp.

6.87.4.5 double Ipopt::LowRankAugSystemSolver::delta_x_ [private]

Most recent value of delta_x from Set method.

Definition at line 128 of file IpLowRankAugSystemSolver.hpp.

6.87.4.6 TaggedObject::Tag Ipopt::LowRankAugSystemSolver::d_s_tag_ [private]

Tag for D_s vector, representing the diagonal matrix D_s.

If D_s has been given to Set as NULL, then this tag is set to 0

Definition at line 132 of file IpLowRankAugSystemSolver.hpp.

6.87.4.7 double Ipopt::LowRankAugSystemSolver::delta_s_ [private]

Most recent value of delta_s from Set method.

Definition at line 134 of file IpLowRankAugSystemSolver.hpp.

6.87.4.8 TaggedObject::Tag Ipopt::LowRankAugSystemSolver::j_c_tag_ [private]

Tag for J_c matrix.

If J_c has been given to Set as NULL, then this tag is set to 0

Definition at line 138 of file IpLowRankAugSystemSolver.hpp.

6.87.4.9 TaggedObject::Tag Ipopt::LowRankAugSystemSolver::d_c_tag_ [private]

Tag for D_c vector, representing the diagonal matrix D_c.

If D_c has been given to Set as NULL, then this tag is set to 0

Definition at line 142 of file IpLowRankAugSystemSolver.hpp.

6.87.4.10 double Ipopt::LowRankAugSystemSolver::delta_c_ [private]

Most recent value of delta_c from Set method.

Definition at line 144 of file IpLowRankAugSystemSolver.hpp.

6.87.4.11 TaggedObject::Tag Ipopt::LowRankAugSystemSolver::j_d_tag_ [private]

Tag for J_d matrix.

If J_d has been given to Set as NULL, then this tag is set to 0

Definition at line 148 of file IpLowRankAugSystemSolver.hpp.

6.87.4.12 TaggedObject::Tag Ipopt::LowRankAugSystemSolver::d_d_tag_ [private]

Tag for D_d vector, representing the diagonal matrix D_d.

If D_d has been given to Set as NULL, then this tag is set to 0

Definition at line 152 of file IpLowRankAugSystemSolver.hpp.

6.87.4.13 double Ipopt::LowRankAugSystemSolver::delta_d_ [private]

Most recent value of delta_d from Set method.

Definition at line 154 of file IpLowRankAugSystemSolver.hpp.

6.87.4.14 bool Ipopt::LowRankAugSystemSolver::first_call_ [private]

Definition at line 160 of file IpLowRankAugSystemSolver.hpp.

6.87.4.15 SmartPtr<DenseGenMatrix> Ipopt::LowRankAugSystemSolver::J1_ [private]

Definition at line 161 of file IpLowRankAugSystemSolver.hpp.

6.87.4.16 SmartPtr<DenseGenMatrix> Ipopt::LowRankAugSystemSolver::J2_ [private]

Definition at line 162 of file IpLowRankAugSystemSolver.hpp.

6.87.4.17 SmartPtr<MultiVectorMatrix> Ipopt::LowRankAugSystemSolver::Vtilde1_ [private]

Definition at line 163 of file IpLowRankAugSystemSolver.hpp.

6.87.4.18 SmartPtr<MultiVectorMatrix> Ipopt::LowRankAugSystemSolver::Utilde2_ [private]

Definition at line 164 of file IpLowRankAugSystemSolver.hpp.

6.87.4.19 SmartPtr<DiagMatrix> Ipopt::LowRankAugSystemSolver::Wdiag_ [private]

Hessian [Matrix](#) passed to the augmented system solver solving the matrix without the low-rank update.

Definition at line 167 of file IpLowRankAugSystemSolver.hpp.

6.87.4.20 SmartPtr<const CompoundVectorSpace> Ipopt::LowRankAugSystemSolver::compound_sol_vecspace_ [private]

[Vector](#) space for Compound vectors that capture the entire right hand side and solution vectors .

Definition at line 170 of file IpLowRankAugSystemSolver.hpp.

6.87.4.21 Index Ipopt::LowRankAugSystemSolver::num_neg_evals_ [private]

Stores the number of negative eigenvalues detected during most recent factorization.

This is what is returned by [NumberOfNegEVals\(\)](#) of this class. It usually is the number of negative eigenvalues returned from the aug_system_solver solve, but if a Cholesky factorization could not be performed, the returned value is one

more than this what the `aug_system_solver` returned.

Definition at line 180 of file `IpLowRankAugSystemSolver.hpp`.

The documentation for this class was generated from the following file:

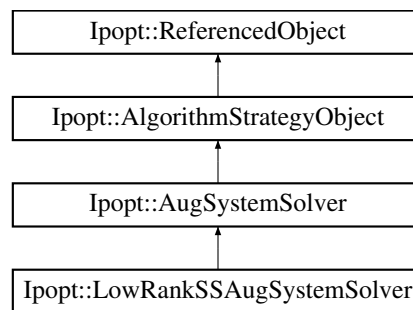
- [Algorithm/IpLowRankAugSystemSolver.hpp](#)

6.88 Ipopt::LowRankSSAugSystemSolver Class Reference

Solver for the augmented system with [LowRankUpdateSymMatrix](#) Hessian matrices.

```
#include <IpLowRankSSAugSystemSolver.hpp>
```

Inheritance diagram for `Ipopt::LowRankSSAugSystemSolver`:



Public Member Functions

- `bool InitializeImpl (const OptionsList &options, const std::string &prefix)`
overloaded from [AlgorithmStrategyObject](#)
- `virtual ESymSolverStatus Solve (const SymMatrix *W, double W_factor, const Vector *D_x, double delta_x, const Vector *D_s, double delta_s, const Matrix *J_c, const Vector *D_c, double delta_c, const Matrix *J_d, const Vector *D_d, double delta_d, const Vector &rhs_x, const Vector &rhs_s, const Vector &rhs_c, const Vector &rhs_d, Vector &sol_x, Vector &sol_s, Vector &sol_c, Vector &sol_d, bool check_NegEVals, Index numberOfNegEVals)`
Set up the augmented system and solve it for a given right hand side.
- `virtual Index NumberOfNegEVals () const`
Number of negative eigenvalues detected during last solve.
- `virtual bool ProvidesInertia () const`
Query whether inertia is computed by linear solver.
- `virtual bool IncreaseQuality ()`
Request to increase quality of solution for next solve.

Constructors/Destructors

- `LowRankSSAugSystemSolver (AugSystemSolver &aug_system_solver, Index max_rank)`
Constructor using an existing augmented system solver.
- `virtual ~LowRankSSAugSystemSolver ()`
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [LowRankSSAugSystemSolver](#) ()
Default constructor.
- [LowRankSSAugSystemSolver](#) (const [LowRankSSAugSystemSolver](#) &)
Copy Constructor.
- void [operator=](#) (const [LowRankSSAugSystemSolver](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus UpdateExtendedData](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d, const [Vector](#) &proto_rhs_x, const [Vector](#) &proto_rhs_s, const [Vector](#) &proto_rhs_c, const [Vector](#) &proto_rhs_d)
Method for updating the factorization, including J1_, J2_, Vtilde1_, Utilde2, Wdiag_, compound_sol_vecspace_.
- bool [AugmentedSystemRequiresChange](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d)
Method that compares the tags of the data for the matrix with those from the previous call.

Private Attributes

- [SmartPtr](#)< [AugSystemSolver](#) > [aug_system_solver_](#)
The augmented system solver object that should be used for the factorization of the augmented system without the low-rank update.
- [Index](#) [max_rank_](#)
Maximal rank of low rank Hessian update.
- bool [first_call_](#)
Flag indicating if this is the first call.
- [Index](#) [num_neg_evals_](#)
Stores the number of negative eigenvalues detected during most recent factorization.

Tags and values to track in order to decide whether the

matrix has to be updated compared to the most recent call of the Set method.

- [TaggedObject::Tag](#) [w_tag_](#)
Tag for W matrix.
- double [w_factor_](#)
Most recent value of W_factor.
- [TaggedObject::Tag](#) [d_x_tag_](#)
Tag for D_x vector, representing the diagonal matrix D_x.
- double [delta_x_](#)
Most recent value of delta_x from Set method.
- [TaggedObject::Tag](#) [d_s_tag_](#)
Tag for D_s vector, representing the diagonal matrix D_s.
- double [delta_s_](#)

- Most recent value of delta_s from Set method.*
- [TaggedObject::Tag j_c_tag_](#)
 - Tag for J_c matrix.*
- [TaggedObject::Tag d_c_tag_](#)
 - Tag for D_c vector, representing the diagonal matrix D_c.*
- double [delta_c_](#)
 - Most recent value of delta_c from Set method.*
- [TaggedObject::Tag j_d_tag_](#)
 - Tag for J_d matrix.*
- [TaggedObject::Tag d_d_tag_](#)
 - Tag for D_d vector, representing the diagonal matrix D_d.*
- double [delta_d_](#)
 - Most recent value of delta_d from Set method.*

Information to be stored in order to resolve for the

same matrix with a different right hand side.

- [SmartPtr< DiagMatrix > Wdiag_](#)
 - Hessian [Matrix](#) passed to the augmented system solver solving the matrix without the low-rank update.*
- [SmartPtr](#)
 - [< ExpandedMultiVectorMatrix > expanded_vu_](#)
 - Artificial rows for Jac_c part for low rank data.*
- [SmartPtr< CompoundMatrix > J_c_ext_](#)
 - Extended Jac_c to include expanded_vu_.*
- [SmartPtr< CompoundVector > D_c_ext_](#)
 - Extended D_c diagonal.*
- [SmartPtr< CompoundVectorSpace > y_c_ext_space_](#)
 - Extended vector space for y_c.*
- [Index negEvalsCorrection_](#)
 - Number of components in V, so that it can be used to correct the inertia.*

Additional Inherited Members

6.88.1 Detailed Description

Solver for the augmented system with [LowRankUpdateSymMatrix](#) Hessian matrices.

This version works with only one backsolve (so it is better for iterative linear solvers), by augmenting the regular augmented system.

Definition at line 27 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.2 Constructor & Destructor Documentation

6.88.2.1 `Ipopt::LowRankSSAugSystemSolver::LowRankSSAugSystemSolver (AugSystemSolver & aug_system_solver, Index max_rank)`

Constructor using an existing augmented system solver.

the `max_rank` argument is the maximal rank that can appear.

6.88.2.2 `virtual Ipopt::LowRankSSAugSystemSolver::~~LowRankSSAugSystemSolver () [virtual]`

Default destructor.

6.88.2.3 Ipopt::LowRankSSAugSystemSolver::LowRankSSAugSystemSolver () [private]

Default constructor.

6.88.2.4 Ipopt::LowRankSSAugSystemSolver::LowRankSSAugSystemSolver (const LowRankSSAugSystemSolver &) [private]

Copy Constructor.

6.88.3 Member Function Documentation

6.88.3.1 bool Ipopt::LowRankSSAugSystemSolver::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AugSystemSolver](#).

6.88.3.2 virtual ESymSolverStatus Ipopt::LowRankSSAugSystemSolver::Solve (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix * J_c, const Vector * D_c, double delta_c, const Matrix * J_d, const Vector * D_d, double delta_d, const Vector & rhs_x, const Vector & rhs_s, const Vector & rhs_c, const Vector & rhs_d, Vector & sol_x, Vector & sol_s, Vector & sol_c, Vector & sol_d, bool check_NegEVals, Index numberOfNegEVals) [virtual]

Set up the augmented system and solve it for a given right hand side.

Reimplemented from [Ipopt::AugSystemSolver](#).

6.88.3.3 virtual Index Ipopt::LowRankSSAugSystemSolver::NumberOfNegEVals () const [virtual]

Number of negative eigenvalues detected during last solve.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see ProvidesInertia).

Implements [Ipopt::AugSystemSolver](#).

6.88.3.4 virtual bool Ipopt::LowRankSSAugSystemSolver::ProvidesInertia () const [virtual]

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::AugSystemSolver](#).

6.88.3.5 virtual bool Ipopt::LowRankSSAugSystemSolver::IncreaseQuality () [virtual]

Request to increase quality of solution for next solve.

Ask underlying linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::AugSystemSolver](#).

6.88.3.6 void Ipopt::LowRankSSAugSystemSolver::operator= (const LowRankSSAugSystemSolver &) [private]

Overloaded Equals Operator.

6.88.3.7 ESymSolverStatus `Ipopt::LowRankSSAugSystemSolver::UpdateExtendedData (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix & J_c, const Vector * D_c, double delta_c, const Matrix & J_d, const Vector * D_d, double delta_d, const Vector & proto_rhs_x, const Vector & proto_rhs_s, const Vector & proto_rhs_c, const Vector & proto_rhs_d) [private]`

Method for updating the factorization, including J1_, J2_, Vtilde1_, Utilde2, Wdiag_, compound_sol_vecspace_.

6.88.3.8 bool `Ipopt::LowRankSSAugSystemSolver::AugmentedSystemRequiresChange (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix & J_c, const Vector * D_c, double delta_c, const Matrix & J_d, const Vector * D_d, double delta_d) [private]`

Method that compares the tags of the data for the matrix with those from the previous call.

Returns true, if there was a change and the factorization has to be updated.

6.88.4 Member Data Documentation

6.88.4.1 SmartPtr<AugSystemSolver> `Ipopt::LowRankSSAugSystemSolver::aug_system_solver_ [private]`

The augmented system solver object that should be used for the factorization of the augmented system without the low-rank update.

Definition at line 115 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.2 Index `Ipopt::LowRankSSAugSystemSolver::max_rank_ [private]`

Maximal rank of low rank Hessian update.

Definition at line 118 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.3 TaggedObject::Tag `Ipopt::LowRankSSAugSystemSolver::w_tag_ [private]`

Tag for W matrix.

If W has been given to Set as NULL, then this tag is set to 0

Definition at line 128 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.4 double `Ipopt::LowRankSSAugSystemSolver::w_factor_ [private]`

Most recent value of W_factor.

Definition at line 130 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.5 TaggedObject::Tag `Ipopt::LowRankSSAugSystemSolver::d_x_tag_ [private]`

Tag for D_x vector, representing the diagonal matrix D_x.

If D_x has been given to Set as NULL, then this tag is set to 0

Definition at line 134 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.6 double `Ipopt::LowRankSSAugSystemSolver::delta_x_ [private]`

Most recent value of delta_x from Set method.

Definition at line 136 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.7 TaggedObject::Tag Ipopt::LowRankSSAugSystemSolver::d_s_tag_ [private]

Tag for D_s vector, representing the diagonal matrix D_s.

If D_s has been given to Set as NULL, then this tag is set to 0

Definition at line 140 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.8 double Ipopt::LowRankSSAugSystemSolver::delta_s_ [private]

Most recent value of delta_s from Set method.

Definition at line 142 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.9 TaggedObject::Tag Ipopt::LowRankSSAugSystemSolver::j_c_tag_ [private]

Tag for J_c matrix.

If J_c has been given to Set as NULL, then this tag is set to 0

Definition at line 146 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.10 TaggedObject::Tag Ipopt::LowRankSSAugSystemSolver::d_c_tag_ [private]

Tag for D_c vector, representing the diagonal matrix D_c.

If D_c has been given to Set as NULL, then this tag is set to 0

Definition at line 150 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.11 double Ipopt::LowRankSSAugSystemSolver::delta_c_ [private]

Most recent value of delta_c from Set method.

Definition at line 152 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.12 TaggedObject::Tag Ipopt::LowRankSSAugSystemSolver::j_d_tag_ [private]

Tag for J_d matrix.

If J_d has been given to Set as NULL, then this tag is set to 0

Definition at line 156 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.13 TaggedObject::Tag Ipopt::LowRankSSAugSystemSolver::d_d_tag_ [private]

Tag for D_d vector, representing the diagonal matrix D_d.

If D_d has been given to Set as NULL, then this tag is set to 0

Definition at line 160 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.14 double Ipopt::LowRankSSAugSystemSolver::delta_d_ [private]

Most recent value of delta_d from Set method.

Definition at line 162 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.15 bool Ipopt::LowRankSSAugSystemSolver::first_call_ [private]

Flag indicating if this is the first call.

Definition at line 166 of file IpLowRankSSAugSystemSolver.hpp.

6.88.4.16 `SmartPtr<DiagMatrix> Ipopt::LowRankSSAugSystemSolver::Wdiag_ [private]`

Hessian [Matrix](#) passed to the augmented system solver solving the matrix without the low-rank update.

Definition at line 173 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.17 `SmartPtr<ExpandedMultiVectorMatrix> Ipopt::LowRankSSAugSystemSolver::expanded_vu_ [private]`

Artificial rows for `Jac_c` part for low rank data.

Definition at line 175 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.18 `SmartPtr<CompoundMatrix> Ipopt::LowRankSSAugSystemSolver::J_c_ext_ [private]`

Extended `Jac_c` to include `expanded_vu_`.

Definition at line 177 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.19 `SmartPtr<CompoundVector> Ipopt::LowRankSSAugSystemSolver::D_c_ext_ [private]`

Extended `D_c` diagonal.

Definition at line 179 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.20 `SmartPtr<CompoundVectorSpace> Ipopt::LowRankSSAugSystemSolver::y_c_ext_space_ [private]`

Extended vector space for `y_c`.

Definition at line 181 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.21 `Index Ipopt::LowRankSSAugSystemSolver::negEvalsCorrection_ [private]`

Number of components in `V`, so that it can be used to correct the inertia.

Definition at line 184 of file `IpLowRankSSAugSystemSolver.hpp`.

6.88.4.22 `Index Ipopt::LowRankSSAugSystemSolver::num_neg_evals_ [private]`

Stores the number of negative eigenvalues detected during most recent factorization.

This is what is returned by `NumberOfNegEvals()` of this class. It usually is the number of negative eigenvalues returned from the `aug_system_solver` solve, but if a Cholesky factorization could not be performed, the returned value is one more than this what the `aug_system_solver` returned.

Definition at line 194 of file `IpLowRankSSAugSystemSolver.hpp`.

The documentation for this class was generated from the following file:

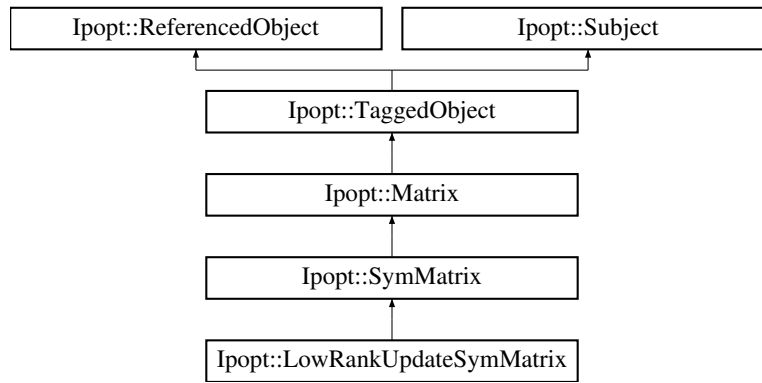
- [Algorithm/IpLowRankSSAugSystemSolver.hpp](#)

6.89 Ipopt::LowRankUpdateSymMatrix Class Reference

Class for symmetric matrices, represented as low-rank updates.

```
#include <IpLowRankUpdateSymMatrix.hpp>
```

Inheritance diagram for `Ipopt::LowRankUpdateSymMatrix`:



Public Member Functions

- void **SetDiag** (const **Vector** &D)
*Method for setting the diagonal elements (as a **Vector**).*
- **SmartPtr**< const **Vector** > **GetDiag** () const
Method for getting the diagonal elements.
- void **SetV** (const **MultiVectorMatrix** &V)
Method for setting the positive low-rank update part.
- **SmartPtr**< const **MultiVectorMatrix** > **GetV** () const
Method for getting the positive low-rank update part.
- void **SetU** (const **MultiVectorMatrix** &U)
Method for setting the negative low-rank update part.
- **SmartPtr**< const **MultiVectorMatrix** > **GetU** () const
Method for getting the negative low-rank update part.
- **SmartPtr**< const **Matrix** > **P_LowRank** () const
Return the expansion matrix to lift the low-rank update to the higher-dimensional space.
- **SmartPtr**< const **VectorSpace** > **LowRankVectorSpace** () const
Return the vector space in with the low-rank update vectors live.
- bool **ReducedDiag** () const
*Flag indicating whether the diagonal term lives in the smaller space (from **P_LowRank**) or in the full space.*

Constructors / Destructors

- **LowRankUpdateSymMatrix** (const **LowRankUpdateSymMatrixSpace** *owner_space)
Constructor, given the corresponding matrix space.
- **~LowRankUpdateSymMatrix** ()
Destructor.

Protected Member Functions

Methods overloaded from matrix

- virtual void **MultiVectorImpl** (**Number** alpha, const **Vector** &x, **Number** beta, **Vector** &y) const
Matrix-vector multiply.
- virtual bool **IsValidNumbersImpl** () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &cols_norms, bool init) const
Since the matrix is symmetric, the row and column max norms are identical.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [LowRankUpdateSymMatrix](#) ()
Default Constructor.
- [LowRankUpdateSymMatrix](#) (const [LowRankUpdateSymMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [LowRankUpdateSymMatrix](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< const [LowRankUpdateSymMatrixSpace](#) > owner_space_
corresponding matrix space
- [SmartPtr](#)< const [Vector](#) > D_
Vector storing the diagonal matrix D.
- [SmartPtr](#)< const [MultiVectorMatrix](#) > V_
Vector storing the positive low-rank update.
- [SmartPtr](#)< const [MultiVectorMatrix](#) > U_
Vector storing the negative low-rank update.

Additional Inherited Members

6.89.1 Detailed Description

Class for symmetric matrices, represented as low-rank updates.

The matrix M is represented as $M = P_LR(D + V V^T - U U^T)P_LR^T$ (if reduced_diag is true), or $M = D + P_LR(V V^T - U U^T)P_LR^T$ (if reduced_diag is false). D is a diagonal matrix, and V and U are MultiVectorMatrices, and P_LR is an [ExpansionMatrix](#). The vectors in the low-rank update (before expansion) live in the LowRankVectorSpace. If P_LR is NULL, P_LR is assumed to be the identity matrix. If V or U is NULL, it is assume to be a matrix of zero columns.

Definition at line 31 of file IpLowRankUpdateSymMatrix.hpp.

6.89.2 Constructor & Destructor Documentation

6.89.2.1 `Ipopt::LowRankUpdateSymMatrix::LowRankUpdateSymMatrix (const LowRankUpdateSymMatrixSpace * owner_space)`

Constructor, given the corresponding matrix space.

6.89.2.2 Ipopt::LowRankUpdateSymMatrix::~~LowRankUpdateSymMatrix ()

Destructor.

6.89.2.3 Ipopt::LowRankUpdateSymMatrix::LowRankUpdateSymMatrix () [private]

Default Constructor.

6.89.2.4 Ipopt::LowRankUpdateSymMatrix::LowRankUpdateSymMatrix (const LowRankUpdateSymMatrix &) [private]

Copy Constructor.

6.89.3 Member Function Documentation

6.89.3.1 void Ipopt::LowRankUpdateSymMatrix::SetDiag (const Vector & D) [inline]

Method for setting the diagonal elements (as a [Vector](#)).

Definition at line 46 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.2 SmartPtr<const Vector> Ipopt::LowRankUpdateSymMatrix::GetDiag () const [inline]

Method for getting the diagonal elements.

Definition at line 53 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.3 void Ipopt::LowRankUpdateSymMatrix::SetV (const MultiVectorMatrix & V) [inline]

Method for setting the positive low-rank update part.

Definition at line 59 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.4 SmartPtr<const MultiVectorMatrix> Ipopt::LowRankUpdateSymMatrix::GetV () const [inline]

Method for getting the positive low-rank update part.

Definition at line 66 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.5 void Ipopt::LowRankUpdateSymMatrix::SetU (const MultiVectorMatrix & U) [inline]

Method for setting the negative low-rank update part.

Definition at line 72 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.6 SmartPtr<const MultiVectorMatrix> Ipopt::LowRankUpdateSymMatrix::GetU () const [inline]

Method for getting the negative low-rank update part.

Definition at line 79 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.7 SmartPtr<const Matrix> Ipopt::LowRankUpdateSymMatrix::P_LowRank () const [inline]

Return the expansion matrix to lift the low-rank update to the higher-dimensional space.

Definition at line 237 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.8 SmartPtr<const VectorSpace> Ipopt::LowRankUpdateSymMatrix::LowRankVectorSpace () const [inline]

Return the vector space in with the low-rank update vectors live.

Definition at line 243 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.9 `bool Ipopt::LowRankUpdateSymMatrix::ReducedDiag () const [inline]`

Flag indicating whether the diagonal term lives in the smaller space (from P_LowRank) or in the full space.

Definition at line 249 of file IpLowRankUpdateSymMatrix.hpp.

6.89.3.10 `virtual void Ipopt::LowRankUpdateSymMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected],[virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.89.3.11 `virtual bool Ipopt::LowRankUpdateSymMatrix::IsValidNumbersImpl () const [protected],[virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.89.3.12 `virtual void Ipopt::LowRankUpdateSymMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const [protected],[virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.89.3.13 `virtual void Ipopt::LowRankUpdateSymMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const [protected],[virtual]`

Since the matrix is symmetric, the row and column max norms are identical.

Reimplemented from [Ipopt::SymMatrix](#).

6.89.3.14 `virtual void Ipopt::LowRankUpdateSymMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const [protected],[virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.89.3.15 `void Ipopt::LowRankUpdateSymMatrix::operator= (const LowRankUpdateSymMatrix &) [private]`

Overloaded Equals Operator.

6.89.4 Member Data Documentation

6.89.4.1 `SmartPointer<const LowRankUpdateSymMatrixSpace> Ipopt::LowRankUpdateSymMatrix::owner_space_ [private]`

corresponding matrix space

Definition at line 138 of file IpLowRankUpdateSymMatrix.hpp.

6.89.4.2 SmartPtr<const Vector> Ipopt::LowRankUpdateSymMatrix::D_ [private]

[Vector](#) storing the diagonal matrix D.

Definition at line 141 of file IpLowRankUpdateSymMatrix.hpp.

6.89.4.3 SmartPtr<const MultiVectorMatrix> Ipopt::LowRankUpdateSymMatrix::V_ [private]

[Vector](#) storing the positive low-rank update.

Definition at line 144 of file IpLowRankUpdateSymMatrix.hpp.

6.89.4.4 SmartPtr<const MultiVectorMatrix> Ipopt::LowRankUpdateSymMatrix::U_ [private]

[Vector](#) storing the negative low-rank update.

Definition at line 147 of file IpLowRankUpdateSymMatrix.hpp.

The documentation for this class was generated from the following file:

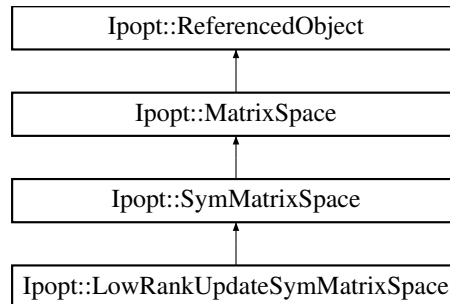
- [LinAlg/IpLowRankUpdateSymMatrix.hpp](#)

6.90 Ipopt::LowRankUpdateSymMatrixSpace Class Reference

This is the matrix space for [LowRankUpdateSymMatrix](#).

```
#include <IpLowRankUpdateSymMatrix.hpp>
```

Inheritance diagram for Ipopt::LowRankUpdateSymMatrixSpace:



Public Member Functions

- virtual [SymMatrix](#) * [MakeNewSymMatrix](#) () const
Overloaded MakeNew method for the [SymMatrixSpace](#) base class.
- [LowRankUpdateSymMatrix](#) * [MakeNewLowRankUpdateSymMatrix](#) () const
Method for creating a new matrix of this specific type.
- [SmartPtr](#)< const [Matrix](#) > [P_LowRank](#) () const
- [SmartPtr](#)< const [VectorSpace](#) > [LowRankVectorSpace](#) () const
- bool [ReducedDiag](#) () const

Constructors / Destructors

- [LowRankUpdateSymMatrixSpace](#) (Index dim, [SmartPtr](#)< const [Matrix](#) > [P_LowRank](#), [SmartPtr](#)< const [VectorSpace](#) > [LowRankVectorSpace](#), bool reduced_diag)

- *Constructor, given the dimension of the matrix.*
• virtual [~LowRankUpdateSymMatrixSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [LowRankUpdateSymMatrixSpace](#) ()
Default Constructor.
- [LowRankUpdateSymMatrixSpace](#) (const [LowRankUpdateSymMatrixSpace](#) &)
Copy Constructor.
- void [operator=](#) (const [LowRankUpdateSymMatrixSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< const [Matrix](#) > [P_LowRank_](#)
Expansion matrix to lift the low-rank approximation into a possibly higher-dimensional space.
- [SmartPtr](#)< const [VectorSpace](#) > [lowrank_vector_space_](#)
[Vector](#) space for the space in which the low-rank approximation lives.
- bool [reduced_diag_](#)
Flag indicating whether the diagonal matrix is nonzero only in the space of V or in the full space.

6.90.1 Detailed Description

This is the matrix space for [LowRankUpdateSymMatrix](#).

Definition at line 151 of file [IpLowRankUpdateSymMatrix.hpp](#).

6.90.2 Constructor & Destructor Documentation

6.90.2.1 `Ipopt::LowRankUpdateSymMatrixSpace::LowRankUpdateSymMatrixSpace (Index dim, SmartPtr< const Matrix > P_LowRank, SmartPtr< const VectorSpace > LowRankVectorSpace, bool reduced_diag) [inline]`

Constructor, given the dimension of the matrix.

Definition at line 157 of file [IpLowRankUpdateSymMatrix.hpp](#).

6.90.2.2 `virtual Ipopt::LowRankUpdateSymMatrixSpace::~~LowRankUpdateSymMatrixSpace () [inline],[virtual]`

Destructor.

Definition at line 171 of file [IpLowRankUpdateSymMatrix.hpp](#).

6.90.2.3 `Ipopt::LowRankUpdateSymMatrixSpace::LowRankUpdateSymMatrixSpace () [private]`

Default Constructor.

6.90.2.4 `Ipopt::LowRankUpdateSymMatrixSpace::LowRankUpdateSymMatrixSpace (const LowRankUpdateSymMatrixSpace &) [private]`

Copy Constructor.

6.90.3 Member Function Documentation

6.90.3.1 `virtual SymMatrix* Ipopt::LowRankUpdateSymMatrixSpace::MakeNewSymMatrix () const [inline], [virtual]`

Overloaded MakeNew method for the [SymMatrixSpace](#) base class.

Implements [Ipopt::SymMatrixSpace](#).

Definition at line 177 of file `IpLowRankUpdateSymMatrix.hpp`.

6.90.3.2 `LowRankUpdateSymMatrix* Ipopt::LowRankUpdateSymMatrixSpace::MakeNewLowRankUpdateSymMatrix () const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 183 of file `IpLowRankUpdateSymMatrix.hpp`.

6.90.3.3 `SmartPtr<const Matrix> Ipopt::LowRankUpdateSymMatrixSpace::P_LowRank () const [inline]`

Definition at line 188 of file `IpLowRankUpdateSymMatrix.hpp`.

6.90.3.4 `SmartPtr<const VectorSpace> Ipopt::LowRankUpdateSymMatrixSpace::LowRankVectorSpace () const [inline]`

Definition at line 193 of file `IpLowRankUpdateSymMatrix.hpp`.

6.90.3.5 `bool Ipopt::LowRankUpdateSymMatrixSpace::ReducedDiag () const [inline]`

Definition at line 198 of file `IpLowRankUpdateSymMatrix.hpp`.

6.90.3.6 `void Ipopt::LowRankUpdateSymMatrixSpace::operator= (const LowRankUpdateSymMatrixSpace &) [private]`

Overloaded Equals Operator.

6.90.4 Member Data Documentation

6.90.4.1 `SmartPtr<const Matrix> Ipopt::LowRankUpdateSymMatrixSpace::P_LowRank_ [private]`

Expansion matrix to lift the low-rank approximation into a possibly higher-dimensional space.

If it is NULL, it is assume that no lift is performed.

Definition at line 225 of file `IpLowRankUpdateSymMatrix.hpp`.

6.90.4.2 `SmartPtr<const VectorSpace> Ipopt::LowRankUpdateSymMatrixSpace::lowrank_vector_space_ [private]`

[Vector](#) space for the space in which the low-rank approximation lives.

Definition at line 229 of file `IpLowRankUpdateSymMatrix.hpp`.

6.90.4.3 `bool Ipopt::LowRankUpdateSymMatrixSpace::reduced_diag_ [private]`

Flag indicating whether the diagonal matrix is nonzero only in the space of V or in the full space.

Definition at line 233 of file `IpLowRankUpdateSymMatrix.hpp`.

The documentation for this class was generated from the following file:

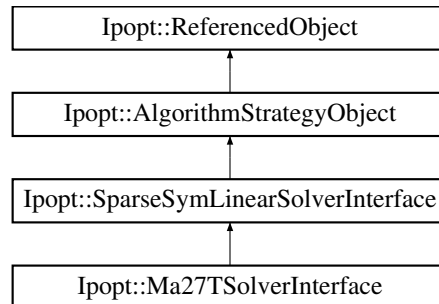
- [LinAlg/IpLowRankUpdateSymMatrix.hpp](#)

6.91 `Ipopt::Ma27TSolverInterface` Class Reference

Interface to the symmetric linear solver MA27, derived from [SparseSymLinearSolverInterface](#).

```
#include <IpMa27TSolverInterface.hpp>
```

Inheritance diagram for `Ipopt::Ma27TSolverInterface`:



Public Member Functions

- `bool InitializeImpl (const OptionsList &options, const std::string &prefix)`
overloaded from [AlgorithmStrategyObject](#)

Constructor/Destructor

- `Ma27TSolverInterface ()`
Constructor.
- `virtual ~Ma27TSolverInterface ()`
Destructor.

Methods for requesting solution of the linear system.

- `virtual ESymSolverStatus InitializeStructure (Index dim, Index nonzeros, const Index *airn, const Index *ajcn)`
Method for initializing internal structures.
- `virtual double * GetValuesArrayPtr ()`
Method returning an internal array into which the nonzero elements (in the same order as airn and ajcn) are to be stored by the calling routine before a call to MultiSolve with a new_matrix=true.
- `virtual ESymSolverStatus MultiSolve (bool new_matrix, const Index *airn, const Index *ajcn, Index nrhs, double *rhs_vals, bool check_NegEvals, Index numberOfNegEvals)`
Solve operation for multiple right hand sides.
- `virtual Index NumberOfNegEvals () const`
Number of negative eigenvalues detected during last factorization.

- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat](#) [MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Static Public Member Functions

- static void [RegisterOptions](#) (SmartPtr< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [Ma27TSolverInterface](#) (const [Ma27TSolverInterface](#) &)
Copy Constructor.
- void [operator=](#) (const [Ma27TSolverInterface](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus](#) [SymbolicFactorization](#) (const [Index](#) *airn, const [Index](#) *ajcn)
Call MA27AD and reserve memory for MA27 data.
- [ESymSolverStatus](#) [Factorization](#) (const [Index](#) *airn, const [Index](#) *ajcn, bool check_NegEVals, [Index](#) number-OfNegEVals)
Call MA27BD to factorize the [Matrix](#).
- [ESymSolverStatus](#) [Backsolve](#) ([Index](#) nrhs, double *rhs_vals)
Call MA27CD to do the backsolve.

Private Attributes

Information about the matrix

- [Index](#) [dim_](#)
Number of rows and columns of the matrix.
- [Index](#) [nonzeros_](#)
Number of nonzeros of the matrix.

Information about most recent factorization/solve

- [Index](#) [negevals_](#)
Number of negative eigenvalues.

Initialization flags

- bool [initialized_](#)

- *Flag indicating if internal data is initialized.*
- bool [pivtol_changed_](#)
Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed.
- bool [refactorize_](#)
Flag that is true if we just requested the values of the matrix again (SYMSOLVER_CALL_AGAIN) and have to factorize again.

Solver specific data/options

- Number [pivtol_](#)
Pivot tolerance.
- Number [pivtolmax_](#)
Maximal pivot tolerance.
- Number [liw_init_factor_](#)
Factor for estimating initial value of liw.
- Number [la_init_factor_](#)
Factor for estimating initial value of la.
- Number [meminc_factor_](#)
Factor for increaseing memory.
- bool [warm_start_same_structure_](#)
*Flag indicating whether the *TNLP* with identical structure has already been solved before.*
- bool [skip_inertia_check_](#)
Flag indicating if the interia is always assumed to be correct.
- bool [ignore_singularity_](#)
Flag indicating if MA27 should continue if a singular matrix is detected, but right hands sides are still accepted.

Data for the linear solver.

Storing factorization and other solver specific data structure.

- [ipfint icntl_](#) [30]
integer control values
- double [cntl_](#) [5]
real control values
- [ipfint liw_](#)
length of integer work space
- [ipfint * iw_](#)
integer work space
- [ipfint * ikeep_](#)
MA27's IKEEP.
- [ipfint nsteps_](#)
MA27's NSTEPS.
- [ipfint maxfrt_](#)
MA27's MAXFRT.
- [ipfint la_](#)
length LA of A
- double * [a_](#)
factor A of matrix
- bool [la_increase_](#)
flag indicating that la should be increased before next factorization
- bool [liw_increase_](#)
flag indicating that liw should be increased before next factorization

Additional Inherited Members

6.91.1 Detailed Description

Interface to the symmetric linear solver MA27, derived from [SparseSymLinearSolverInterface](#).

Definition at line 19 of file IpMa27TSolverInterface.hpp.

6.91.2 Constructor & Destructor Documentation

6.91.2.1 Ipopt::Ma27TSolverInterface::Ma27TSolverInterface ()

Constructor.

6.91.2.2 virtual Ipopt::Ma27TSolverInterface::~~Ma27TSolverInterface () [virtual]

Destructor.

6.91.2.3 Ipopt::Ma27TSolverInterface::Ma27TSolverInterface (const Ma27TSolverInterface &) [private]

Copy Constructor.

6.91.3 Member Function Documentation

6.91.3.1 bool Ipopt::Ma27TSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.91.3.2 virtual ESymSolverStatus Ipopt::Ma27TSolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * airn, const Index * ajcn) [virtual]

Method for initializing internal stuctures.

Here, ndim gives the number of rows and columns of the matrix, nonzeros give the number of nonzero elements, and airn and ajcn give the positions of the nonzero elements.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.91.3.3 virtual double* Ipopt::Ma27TSolverInterface::GetValuesArrayPtr () [virtual]

Method returning an internal array into which the nonzero elements (in the same order as airn and ajcn) are to be stored by the calling routine before a call to MultiSolve with a new_matrix=true.

The returned array must have space for at least nonzero elements.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.91.3.4 virtual ESymSolverStatus Ipopt::Ma27TSolverInterface::MultiSolve (bool new_matrix, const Index * airn, const Index * ajcn, Index nrhs, double * rhs_vals, bool check_NegEVals, Index numberOfNegEVals) [virtual]

Solve operation for multiple right hand sides.

Overloaded from [SparseSymLinearSolverInterface](#).

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.91.3.5 `virtual Index Ipopt::Ma27TSolverInterface::NumberOfNegEVals () const [virtual]`

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see `ProvidesInertia`).

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.91.3.6 `virtual bool Ipopt::Ma27TSolverInterface::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

Ask linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.91.3.7 `virtual bool Ipopt::Ma27TSolverInterface::ProvidesInertia () const [inline],[virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 86 of file `IpMa27TSolverInterface.hpp`.

6.91.3.8 `EMatrixFormat Ipopt::Ma27TSolverInterface::MatrixFormat () const [inline],[virtual]`

Query of requested matrix type that the linear solver understands.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 93 of file `IpMa27TSolverInterface.hpp`.

6.91.3.9 `static void Ipopt::Ma27TSolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for `IpoptType`.

6.91.3.10 `void Ipopt::Ma27TSolverInterface::operator= (const Ma27TSolverInterface &) [private]`

Overloaded Equals Operator.

6.91.3.11 `ESymSolverStatus Ipopt::Ma27TSolverInterface::SymbolicFactorization (const Index * airn, const Index * ajcn) [private]`

Call MA27AD and reserve memory for MA27 data.

Reserve memory for `iw_` and `ikeep_`, call MA27AD to perform symbolic manipulations, and reserve all the remaining data memory

6.91.3.12 `ESymSolverStatus Ipopt::Ma27TSolverInterface::Factorization (const Index * airn, const Index * ajcn, bool check_NegEVals, Index numberOfNegEVals) [private]`

Call MA27BD to factorize the [Matrix](#).

It is assumed that the first `nonzeros_` element of `a_` contain the values of the matrix to be factorized.

6.91.3.13 **ESymSolverStatus** Ipopt::Ma27TSolverInterface::Backsolve (Index *nrhs*, double * *rhs_vals*) [private]

Call MA27CD to do the backsolve.

6.91.4 Member Data Documentation

6.91.4.1 **Index** Ipopt::Ma27TSolverInterface::dim_ [private]

Number of rows and columns of the matrix.

Definition at line 123 of file IpMa27TSolverInterface.hpp.

6.91.4.2 **Index** Ipopt::Ma27TSolverInterface::nonzeros_ [private]

Number of nonzeros of the matrix.

Definition at line 126 of file IpMa27TSolverInterface.hpp.

6.91.4.3 **Index** Ipopt::Ma27TSolverInterface::negevals_ [private]

Number of negative eigenvalues.

Definition at line 132 of file IpMa27TSolverInterface.hpp.

6.91.4.4 **bool** Ipopt::Ma27TSolverInterface::initialized_ [private]

Flag indicating if internal data is initialized.

For initialization, this object needs to have seen a matrix

Definition at line 139 of file IpMa27TSolverInterface.hpp.

6.91.4.5 **bool** Ipopt::Ma27TSolverInterface::pivtol_changed_ [private]

Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed.

Definition at line 142 of file IpMa27TSolverInterface.hpp.

6.91.4.6 **bool** Ipopt::Ma27TSolverInterface::refactorize_ [private]

Flag that is true if we just requested the values of the matrix again (SYMSOLVER_CALL_AGAIN) and have to factorize again.

Definition at line 146 of file IpMa27TSolverInterface.hpp.

6.91.4.7 **Number** Ipopt::Ma27TSolverInterface::pivtol_ [private]

Pivot tolerance.

Definition at line 152 of file IpMa27TSolverInterface.hpp.

6.91.4.8 **Number** Ipopt::Ma27TSolverInterface::pivtolmax_ [private]

Maximal pivot tolerance.

Definition at line 155 of file IpMa27TSolverInterface.hpp.

6.91.4.9 **Number** Ipopt::Ma27TSolverInterface::liw_init_factor_ [private]

Factor for estimating initial value of liw.

Definition at line 158 of file IpMa27TSolverInterface.hpp.

6.91.4.10 `Number Ipopt::Ma27TSolverInterface::la_init_factor_ [private]`

Factor for estimating initial value of la.

Definition at line 160 of file IpMa27TSolverInterface.hpp.

6.91.4.11 `Number Ipopt::Ma27TSolverInterface::meminc_factor_ [private]`

Factor for increaseing memory.

Definition at line 162 of file IpMa27TSolverInterface.hpp.

6.91.4.12 `bool Ipopt::Ma27TSolverInterface::warm_start_same_structure_ [private]`

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 165 of file IpMa27TSolverInterface.hpp.

6.91.4.13 `bool Ipopt::Ma27TSolverInterface::skip_inertia_check_ [private]`

Flag indicating if the interia is always assumed to be correct.

Definition at line 168 of file IpMa27TSolverInterface.hpp.

6.91.4.14 `bool Ipopt::Ma27TSolverInterface::ignore_singularity_ [private]`

Flag indicating if MA27 should continue if a singular matrix

is detected, but right hands sides are still accepted.

Definition at line 171 of file IpMa27TSolverInterface.hpp.

6.91.4.15 `ipfint Ipopt::Ma27TSolverInterface::icntl[30] [private]`

integer control values

Definition at line 179 of file IpMa27TSolverInterface.hpp.

6.91.4.16 `double Ipopt::Ma27TSolverInterface::cntl[5] [private]`

real control values

Definition at line 181 of file IpMa27TSolverInterface.hpp.

6.91.4.17 `ipfint Ipopt::Ma27TSolverInterface::liw_ [private]`

length of integer work space

Definition at line 184 of file IpMa27TSolverInterface.hpp.

6.91.4.18 `ipfint* Ipopt::Ma27TSolverInterface::iw_ [private]`

integer work space

Definition at line 186 of file IpMa27TSolverInterface.hpp.

6.91.4.19 `ipfint* Ipopt::Ma27TSolverInterface::ikeep_ [private]`

MA27's IKEEP.

Definition at line 189 of file IpMa27TSolverInterface.hpp.

6.91.4.20 `ipfint Ipopt::Ma27TSolverInterface::nsteps_ [private]`

MA27's NSTEPS.

Definition at line 191 of file IpMa27TSolverInterface.hpp.

6.91.4.21 `ipfint Ipopt::Ma27TSolverInterface::maxfirt_ [private]`

MA27's MAXFRT.

Definition at line 193 of file IpMa27TSolverInterface.hpp.

6.91.4.22 `ipfint Ipopt::Ma27TSolverInterface::la_ [private]`

length LA of A

Definition at line 196 of file IpMa27TSolverInterface.hpp.

6.91.4.23 `double* Ipopt::Ma27TSolverInterface::a_ [private]`

factor A of matrix

Definition at line 198 of file IpMa27TSolverInterface.hpp.

6.91.4.24 `bool Ipopt::Ma27TSolverInterface::la_increase_ [private]`

flag indicating that la should be increased before next factorization

Definition at line 202 of file IpMa27TSolverInterface.hpp.

6.91.4.25 `bool Ipopt::Ma27TSolverInterface::liw_increase_ [private]`

flag indicating that liw should be increased before next factorization

Definition at line 205 of file IpMa27TSolverInterface.hpp.

The documentation for this class was generated from the following file:

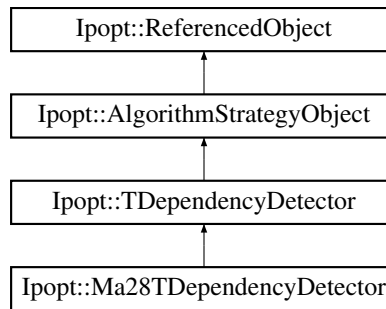
- Algorithm/LinearSolvers/[IpMa27TSolverInterface.hpp](#)

6.92 Ipopt::Ma28TDependencyDetector Class Reference

Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

```
#include <IpMa28TDependencyDetector.hpp>
```

Inheritance diagram for Ipopt::Ma28TDependencyDetector:



Public Member Functions

- virtual bool `InitializeImpl` (const `OptionsList` &options, const std::string &prefix)
Has to be called to initialize and reset these objects.
- virtual bool `DetermineDependentRows` (Index n_rows, Index n_cols, Index n_jac_nz, Number *jac_c_vals, Index *jac_c_iRow, Index *jac_c_jCol, std::list< Index > &c_deps)
Method determining the number of linearly dependent rows in the matrix and the indices of those rows.

Constructor/Destructor

- `Ma28TDependencyDetector` ()
- virtual `~Ma28TDependencyDetector` ()

Static Public Member Functions

- static void `RegisterOptions` (SmartPtr< `RegisteredOptions` > roptions)
This must be called to make the options for this class known.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `Ma28TDependencyDetector` (const `Ma28TDependencyDetector` &)
Copy Constructor.
- void `operator=` (const `Ma28TDependencyDetector` &)
Overloaded Equals Operator.

Private Attributes

- SmartPtr< const `Journalist` > jnlst_

Algorithmic parameters

- Number `ma28_pivotl_`
Pivot tolerance for MA28.

Additional Inherited Members

6.92.1 Detailed Description

Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

Definition at line 19 of file IpMa28TDependencyDetector.hpp.

6.92.2 Constructor & Destructor Documentation

6.92.2.1 Ipopt::Ma28TDependencyDetector::Ma28TDependencyDetector ()

6.92.2.2 virtual Ipopt::Ma28TDependencyDetector::~~Ma28TDependencyDetector () [inline],[virtual]

Definition at line 26 of file IpMa28TDependencyDetector.hpp.

6.92.2.3 Ipopt::Ma28TDependencyDetector::Ma28TDependencyDetector (const Ma28TDependencyDetector &)
[private]

Copy Constructor.

6.92.3 Member Function Documentation

6.92.3.1 virtual bool Ipopt::Ma28TDependencyDetector::InitializeImpl (const OptionsList & options, const std::string & prefix)
[virtual]

Has to be called to initialize and reset these objects.

Implements [Ipopt::TDependencyDetector](#).

6.92.3.2 virtual bool Ipopt::Ma28TDependencyDetector::DetermineDependentRows (Index n_rows, Index n_cols, Index n_jac_nz, Number * jac_c_vals, Index * jac_c_iRow, Index * jac_c_jCol, std::list< Index > & c_deps)
[virtual]

Method determining the number of linearly dependent rows in the matrix and the indices of those rows.

We assume that the matrix is available in "Triplet" format (MA28 format), and that the arrays given to this method can be modified internally, i.e., they are not used by the calling program anymore after this call. This method returns false if there was a problem with the underlying linear solver.

Implements [Ipopt::TDependencyDetector](#).

6.92.3.3 static void Ipopt::Ma28TDependencyDetector::RegisterOptions (SmartPtr< RegisteredOptions > roptions)
[static]

This must be called to make the options for this class known.

6.92.3.4 void Ipopt::Ma28TDependencyDetector::operator= (const Ma28TDependencyDetector &) [private]

Overloaded Equals Operator.

6.92.4 Member Data Documentation

6.92.4.1 `SmartPtr<const Journalist> Ipopt::Ma28TDependencyDetector::jnlst_` [private]

Definition at line 69 of file `IpMa28TDependencyDetector.hpp`.

6.92.4.2 `Number Ipopt::Ma28TDependencyDetector::ma28_pivotl_` [private]

Pivot tolerance for MA28.

Definition at line 74 of file `IpMa28TDependencyDetector.hpp`.

The documentation for this class was generated from the following file:

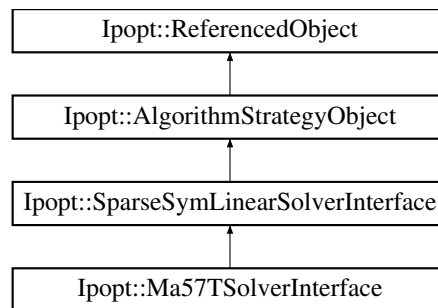
- [Algorithm/LinearSolvers/IpMa28TDependencyDetector.hpp](#)

6.93 `Ipopt::Ma57TSolverInterface` Class Reference

Interface to the symmetric linear solver MA57, derived from [SparseSymLinearSolverInterface](#).

```
#include <IpMa57TSolverInterface.hpp>
```

Inheritance diagram for `Ipopt::Ma57TSolverInterface`:



Public Member Functions

- `bool InitializeImpl` (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Constructor/Destructor

- [Ma57TSolverInterface](#) ()
Constructor.
- virtual `~Ma57TSolverInterface` ()
Destructor.

Methods for requesting solution of the linear system.

- virtual `ESymSolverStatus InitializeStructure` ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *airn, const [Index](#) *ajcn)
Method for initializing internal stuctures.
- virtual `double * GetValuesArrayPtr` ()
Method returng an internal array into which the nonzero elements (in the same order as airn and ajcn) are to be stored by the calling routine before a call to MultiSolve with a new_matrix=true.
- virtual `ESymSolverStatus MultiSolve` (bool new_matrix, const [Index](#) *airn, const [Index](#) *ajcn, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)

- virtual [Index NumberOfNegEvals](#) () const
Number of negative eigenvalues detected during last factorization.
- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat](#) [MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Static Public Member Functions

- static void [RegisterOptions](#) (SmartPtr< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [Ma57TSolverInterface](#) (const [Ma57TSolverInterface](#) &)
Copy Constructor.
- void [operator=](#) (const [Ma57TSolverInterface](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus SymbolicFactorization](#) (const [Index](#) *airn, const [Index](#) *ajcn)
Call MA57AD and reserve memory for MA57 data.
- [ESymSolverStatus Factorization](#) (const [Index](#) *airn, const [Index](#) *ajcn, bool check_NegEvals, [Index](#) number-OfNegEvals)
Call MA57BD to factorize the [Matrix](#).
- [ESymSolverStatus Backsolve](#) ([Index](#) nrhs, double *rhs_vals)
Call MA57CD to do the backsolve.

Private Attributes

Information about the matrix

- [Index dim_](#)
Number of rows and columns of the matrix.
- [Index nonzeros_](#)
Number of nonzeros of the matrix.

Information about most recent factorization/solve

- [Index negevals_](#)
Number of negative eigenvalues.

Initialization flags

- bool `initialized_`
Flag indicating if internal data is initialized.
- bool `pivtol_changed_`
Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed.
- bool `refactorize_`
Flag that is true if we just requested the values of the matrix again (SYMSOLVER_CALL_AGAIN) and have to factorize again.

Solver specific data/options

- Number `pivtol_`
Pivot tolerance.
- Number `pivtolmax_`
Maximal pivot tolerance.
- Number `ma57_pre_alloc_`
Factor for estimating initial size of work arrays.
- bool `warm_start_same_structure_`
*Flag indicating whether the *TNLP* with identical structure has already been solved before.*

Data for the linear solver.

Storing factorization and other solver specific data structure.

- double `wd_cntl_` [5]
- `ma57int` `wd_icntl_` [20]
- `ma57int` `wd_info_` [40]
- double `wd_rinfo_` [20]
- `ma57int` `wd_lkeep_`
- `ma57int` * `wd_keep_`
- `ma57int` * `wd_iwork_`
- double * `wd_fact_`
- `ma57int` `wd_lfact_`
- `ma57int` * `wd_ifact_`
- `ma57int` `wd_lifact_`
- double * `a_`
factor A of matrix

Additional Inherited Members

6.93.1 Detailed Description

Interface to the symmetric linear solver MA57, derived from [SparseSymLinearSolverInterface](#).

Definition at line 27 of file `IpMa57TSolverInterface.hpp`.

6.93.2 Constructor & Destructor Documentation

6.93.2.1 `Ipopt::Ma57TSolverInterface::Ma57TSolverInterface ()`

Constructor.

6.93.2.2 `virtual Ipopt::Ma57TSolverInterface::~Ma57TSolverInterface () [virtual]`

Destructor.

6.93.2.3 Ipopt::Ma57TSolverInterface::Ma57TSolverInterface (const Ma57TSolverInterface &) [private]

Copy Constructor.

6.93.3 Member Function Documentation

6.93.3.1 bool Ipopt::Ma57TSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.93.3.2 virtual ESymSolverStatus Ipopt::Ma57TSolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * airn, const Index * ajcn) [virtual]

Method for initializing internal structures.

Here, ndim gives the number of rows and columns of the matrix, nonzeros give the number of nonzero elements, and airn and ajcn give the positions of the nonzero elements.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.93.3.3 virtual double* Ipopt::Ma57TSolverInterface::GetValuesArrayPtr () [virtual]

Method returning an internal array into which the nonzero elements (in the same order as airn and ajcn) are to be stored by the calling routine before a call to MultiSolve with a new_matrix=true.

The returned array must have space for at least nonzero elements.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.93.3.4 virtual ESymSolverStatus Ipopt::Ma57TSolverInterface::MultiSolve (bool new_matrix, const Index * airn, const Index * ajcn, Index nrhs, double * rhs_vals, bool check_NegEvals, Index numberOfNegEvals) [virtual]

Solve operation for multiple right hand sides.

Overloaded from [SparseSymLinearSolverInterface](#).

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.93.3.5 virtual Index Ipopt::Ma57TSolverInterface::NumberOfNegEvals () const [virtual]

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see ProvidesInertia).

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.93.3.6 virtual bool Ipopt::Ma57TSolverInterface::IncreaseQuality () [virtual]

Request to increase quality of solution for next solve.

Ask linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.93.3.7 `virtual bool Ipopt::Ma57TSolverInterface::ProvidesInertia () const [inline],[virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 96 of file IpMa57TSolverInterface.hpp.

6.93.3.8 `EMatrixFormat Ipopt::Ma57TSolverInterface::MatrixFormat () const [inline],[virtual]`

Query of requested matrix type that the linear solver understands.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 103 of file IpMa57TSolverInterface.hpp.

6.93.3.9 `static void Ipopt::Ma57TSolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > options) [static]`

Methods for IpoptType.

6.93.3.10 `void Ipopt::Ma57TSolverInterface::operator= (const Ma57TSolverInterface &) [private]`

Overloaded Equals Operator.

6.93.3.11 `ESymSolverStatus Ipopt::Ma57TSolverInterface::SymbolicFactorization (const Index * airn, const Index * ajcn) [private]`

Call MA57AD and reserve memory for MA57 data.

Reserve memory for iw_ and ikeep_, call MA57AD to perform symbolic manipulations, and reserve all the remaining data memory

6.93.3.12 `ESymSolverStatus Ipopt::Ma57TSolverInterface::Factorization (const Index * airn, const Index * ajcn, bool check_NegEVals, Index numberOfNegEVals) [private]`

Call MA57BD to factorize the [Matrix](#).

It is assumed that the first nonzeros_ element of a_ contain the values of the matrix to be factorized.

6.93.3.13 `ESymSolverStatus Ipopt::Ma57TSolverInterface::Backsolve (Index nrhs, double * rhs_vals) [private]`

Call MA57CD to do the backsolve.

6.93.4 Member Data Documentation

6.93.4.1 `Index Ipopt::Ma57TSolverInterface::dim_ [private]`

Number of rows and columns of the matrix.

Definition at line 133 of file IpMa57TSolverInterface.hpp.

6.93.4.2 `Index Ipopt::Ma57TSolverInterface::nonzeros_ [private]`

Number of nonzeros of the matrix.

Definition at line 136 of file IpMa57TSolverInterface.hpp.

6.93.4.3 Index Ipopt::Ma57TSolverInterface::negevals_ [private]

Number of negative eigenvalues.

Definition at line 142 of file IpMa57TSolverInterface.hpp.

6.93.4.4 bool Ipopt::Ma57TSolverInterface::initialized_ [private]

Flag indicating if internal data is initialized.

For initialization, this object needs to have seen a matrix

Definition at line 149 of file IpMa57TSolverInterface.hpp.

6.93.4.5 bool Ipopt::Ma57TSolverInterface::pivtol_changed_ [private]

Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed.

Definition at line 152 of file IpMa57TSolverInterface.hpp.

6.93.4.6 bool Ipopt::Ma57TSolverInterface::refactorize_ [private]

Flag that is true if we just requested the values of the matrix again (SYMSOLVER_CALL_AGAIN) and have to factorize again.

Definition at line 156 of file IpMa57TSolverInterface.hpp.

6.93.4.7 Number Ipopt::Ma57TSolverInterface::pivtol_ [private]

Pivot tolerance.

Definition at line 162 of file IpMa57TSolverInterface.hpp.

6.93.4.8 Number Ipopt::Ma57TSolverInterface::pivtolmax_ [private]

Maximal pivot tolerance.

Definition at line 164 of file IpMa57TSolverInterface.hpp.

6.93.4.9 Number Ipopt::Ma57TSolverInterface::ma57_pre_alloc_ [private]

Factor for estimating initial size of work arrays.

Definition at line 166 of file IpMa57TSolverInterface.hpp.

6.93.4.10 bool Ipopt::Ma57TSolverInterface::warm_start_same_structure_ [private]

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 169 of file IpMa57TSolverInterface.hpp.

6.93.4.11 double Ipopt::Ma57TSolverInterface::wd_cntl_[5] [private]

Definition at line 176 of file IpMa57TSolverInterface.hpp.

6.93.4.12 ma57int Ipopt::Ma57TSolverInterface::wd_icntl_[20] [private]

Definition at line 177 of file IpMa57TSolverInterface.hpp.

6.93.4.13 ma57int Ipopt::Ma57TSolverInterface::wd_info_[40] [private]

Definition at line 179 of file IpMa57TSolverInterface.hpp.

6.93.4.14 `double Ipopt::Ma57TSolverInterface::wd_rinfo_[20]` [private]

Definition at line 180 of file `IpMa57TSolverInterface.hpp`.

6.93.4.15 `ma57int Ipopt::Ma57TSolverInterface::wd_ikkeep_` [private]

Definition at line 182 of file `IpMa57TSolverInterface.hpp`.

6.93.4.16 `ma57int* Ipopt::Ma57TSolverInterface::wd_ikkeep_` [private]

Definition at line 183 of file `IpMa57TSolverInterface.hpp`.

6.93.4.17 `ma57int* Ipopt::Ma57TSolverInterface::wd_iwork_` [private]

Definition at line 185 of file `IpMa57TSolverInterface.hpp`.

6.93.4.18 `double* Ipopt::Ma57TSolverInterface::wd_fact_` [private]

Definition at line 187 of file `IpMa57TSolverInterface.hpp`.

6.93.4.19 `ma57int Ipopt::Ma57TSolverInterface::wd_lfact_` [private]

Definition at line 188 of file `IpMa57TSolverInterface.hpp`.

6.93.4.20 `ma57int* Ipopt::Ma57TSolverInterface::wd_ifact_` [private]

Definition at line 189 of file `IpMa57TSolverInterface.hpp`.

6.93.4.21 `ma57int Ipopt::Ma57TSolverInterface::wd_lifact_` [private]

Definition at line 190 of file `IpMa57TSolverInterface.hpp`.

6.93.4.22 `double* Ipopt::Ma57TSolverInterface::a_` [private]

factor A of matrix

Definition at line 194 of file `IpMa57TSolverInterface.hpp`.

The documentation for this class was generated from the following file:

- Algorithm/LinearSolvers/[IpMa57TSolverInterface.hpp](#)

6.94 `ma77_control_d` Struct Reference

```
#include <hsl_ma77d.h>
```

Public Attributes

- int [f_arrays](#)
- int [print_level](#)
- int [unit_diagnostics](#)
- int [unit_error](#)
- int [unit_warning](#)
- int [bits](#)
- int [buffer_lpage](#) [2]

- int [buffer_npage](#) [2]
- long int [file_size](#)
- long int [maxstore](#)
- long int [storage](#) [3]
- int [nemin](#)
- int [maxit](#)
- int [infnorm](#)
- [ma77pkgtype_d_thresh](#)
- int [nb54](#)
- int [action](#)
- [ma77pkgtype_d_multiplier](#)
- int [nb64](#)
- int [nbi](#)
- [ma77pkgtype_d_small](#)
- [ma77pkgtype_d_static_](#)
- long int [storage_indef](#)
- [ma77pkgtype_d_u](#)
- [ma77pkgtype_d_u_min](#)
- [ma77pkgtype_d_consist_tol](#)
- int [ispare](#) [5]
- long int [lspare](#) [5]
- [ma77pkgtype_d_rspare](#) [5]

6.94.1 Detailed Description

Definition at line 38 of file `hsl_ma77d.h`.

6.94.2 Member Data Documentation

6.94.2.1 int ma77_control_d::f_arrays

Definition at line 42 of file `hsl_ma77d.h`.

6.94.2.2 int ma77_control_d::print_level

Definition at line 46 of file `hsl_ma77d.h`.

6.94.2.3 int ma77_control_d::unit_diagnostics

Definition at line 47 of file `hsl_ma77d.h`.

6.94.2.4 int ma77_control_d::unit_error

Definition at line 49 of file `hsl_ma77d.h`.

6.94.2.5 int ma77_control_d::unit_warning

Definition at line 51 of file `hsl_ma77d.h`.

6.94.2.6 int ma77_control_d::bits

Definition at line 55 of file `hsl_ma77d.h`.

6.94.2.7 `int ma77_control_d::buffer_lpage[2]`

Definition at line 56 of file `hsl_ma77d.h`.

6.94.2.8 `int ma77_control_d::buffer_npage[2]`

Definition at line 57 of file `hsl_ma77d.h`.

6.94.2.9 `long int ma77_control_d::file_size`

Definition at line 58 of file `hsl_ma77d.h`.

6.94.2.10 `long int ma77_control_d::maxstore`

Definition at line 59 of file `hsl_ma77d.h`.

6.94.2.11 `long int ma77_control_d::storage[3]`

Definition at line 60 of file `hsl_ma77d.h`.

6.94.2.12 `int ma77_control_d::nemin`

Definition at line 63 of file `hsl_ma77d.h`.

6.94.2.13 `int ma77_control_d::maxit`

Definition at line 67 of file `hsl_ma77d.h`.

6.94.2.14 `int ma77_control_d::infnorm`

Definition at line 68 of file `hsl_ma77d.h`.

6.94.2.15 `ma77pkgtype_d ma77_control_d::thresh`

Definition at line 69 of file `hsl_ma77d.h`.

6.94.2.16 `int ma77_control_d::nb54`

Definition at line 72 of file `hsl_ma77d.h`.

6.94.2.17 `int ma77_control_d::action`

Definition at line 75 of file `hsl_ma77d.h`.

6.94.2.18 `ma77pkgtype_d ma77_control_d::multiplier`

Definition at line 77 of file `hsl_ma77d.h`.

6.94.2.19 `int ma77_control_d::nb64`

Definition at line 78 of file `hsl_ma77d.h`.

6.94.2.20 `int ma77_control_d::nbi`

Definition at line 79 of file `hsl_ma77d.h`.

6.94.2.21 ma77pkgtype_d_ma77_control_d::small

Definition at line 80 of file hsl_ma77d.h.

6.94.2.22 ma77pkgtype_d_ma77_control_d::static_

Definition at line 81 of file hsl_ma77d.h.

6.94.2.23 long int ma77_control_d::storage_indef

Definition at line 82 of file hsl_ma77d.h.

6.94.2.24 ma77pkgtype_d_ma77_control_d::u

Definition at line 83 of file hsl_ma77d.h.

6.94.2.25 ma77pkgtype_d_ma77_control_d::umin

Definition at line 84 of file hsl_ma77d.h.

6.94.2.26 ma77pkgtype_d_ma77_control_d::consist_tol

Definition at line 88 of file hsl_ma77d.h.

6.94.2.27 int ma77_control_d::ispare[5]

Definition at line 91 of file hsl_ma77d.h.

6.94.2.28 long int ma77_control_d::lspare[5]

Definition at line 91 of file hsl_ma77d.h.

6.94.2.29 ma77pkgtype_d_ma77_control_d::rspare[5]

Definition at line 91 of file hsl_ma77d.h.

The documentation for this struct was generated from the following file:

- Algorithm/LinearSolvers/[hsl_ma77d.h](#)

6.95 ma77_info_d Struct Reference

```
#include <hsl_ma77d.h>
```

Public Attributes

- [ma77pkgtype_d_detlog](#)
- int [detsign](#)
- int [flag](#)
- int [iostat](#)
- int [matrix_dup](#)
- int [matrix_rank](#)
- int [matrix_outrange](#)
- int [maxdepth](#)
- int [maxfront](#)

- long int [minstore](#)
- int [ndelay](#)
- long int [nfactor](#)
- long int [nflops](#)
- int [niter](#)
- int [nsup](#)
- int [num_neg](#)
- int [num_nothresh](#)
- int [num_perturbed](#)
- int [ntwo](#)
- int [stat](#)
- int [index](#) [4]
- long int [nio_read](#) [2]
- long int [nio_write](#) [2]
- long int [nwd_read](#) [2]
- long int [nwd_write](#) [2]
- int [num_file](#) [4]
- long int [storage](#) [4]
- int [tree_nodes](#)
- int [unit_restart](#)
- int [unused](#)
- [ma77pkgtype_d_usmall](#)
- int [lspare](#) [5]
- long int [lspare](#) [5]
- [ma77pkgtype_d_rspare](#) [5]

6.95.1 Detailed Description

Definition at line 98 of file [hsl_ma77d.h](#).

6.95.2 Member Data Documentation

6.95.2.1 [ma77pkgtype_d_ma77_info_d::detlog](#)

Definition at line 99 of file [hsl_ma77d.h](#).

6.95.2.2 [int ma77_info_d::detsign](#)

Definition at line 100 of file [hsl_ma77d.h](#).

6.95.2.3 [int ma77_info_d::flag](#)

Definition at line 101 of file [hsl_ma77d.h](#).

6.95.2.4 [int ma77_info_d::iostat](#)

Definition at line 102 of file [hsl_ma77d.h](#).

6.95.2.5 [int ma77_info_d::matrix_dup](#)

Definition at line 103 of file [hsl_ma77d.h](#).

6.95.2.6 int ma77_info_d::matrix_rank

Definition at line 104 of file hsl_ma77d.h.

6.95.2.7 int ma77_info_d::matrix_outrange

Definition at line 105 of file hsl_ma77d.h.

6.95.2.8 int ma77_info_d::maxdepth

Definition at line 106 of file hsl_ma77d.h.

6.95.2.9 int ma77_info_d::maxfront

Definition at line 107 of file hsl_ma77d.h.

6.95.2.10 long int ma77_info_d::minstore

Definition at line 108 of file hsl_ma77d.h.

6.95.2.11 int ma77_info_d::ndelay

Definition at line 109 of file hsl_ma77d.h.

6.95.2.12 long int ma77_info_d::nfactor

Definition at line 110 of file hsl_ma77d.h.

6.95.2.13 long int ma77_info_d::nflops

Definition at line 111 of file hsl_ma77d.h.

6.95.2.14 int ma77_info_d::niter

Definition at line 112 of file hsl_ma77d.h.

6.95.2.15 int ma77_info_d::nsup

Definition at line 113 of file hsl_ma77d.h.

6.95.2.16 int ma77_info_d::num_neg

Definition at line 114 of file hsl_ma77d.h.

6.95.2.17 int ma77_info_d::num_nothresh

Definition at line 115 of file hsl_ma77d.h.

6.95.2.18 int ma77_info_d::num_perturbed

Definition at line 116 of file hsl_ma77d.h.

6.95.2.19 int ma77_info_d::ntwo

Definition at line 117 of file hsl_ma77d.h.

6.95.2.20 int ma77_info_d::stat

Definition at line 118 of file hsl_ma77d.h.

6.95.2.21 int ma77_info_d::index[4]

Definition at line 119 of file hsl_ma77d.h.

6.95.2.22 long int ma77_info_d::nio_read[2]

Definition at line 120 of file hsl_ma77d.h.

6.95.2.23 long int ma77_info_d::nio_write[2]

Definition at line 121 of file hsl_ma77d.h.

6.95.2.24 long int ma77_info_d::nwd_read[2]

Definition at line 122 of file hsl_ma77d.h.

6.95.2.25 long int ma77_info_d::nwd_write[2]

Definition at line 123 of file hsl_ma77d.h.

6.95.2.26 int ma77_info_d::num_file[4]

Definition at line 124 of file hsl_ma77d.h.

6.95.2.27 long int ma77_info_d::storage[4]

Definition at line 125 of file hsl_ma77d.h.

6.95.2.28 int ma77_info_d::tree_nodes

Definition at line 126 of file hsl_ma77d.h.

6.95.2.29 int ma77_info_d::unit_restart

Definition at line 127 of file hsl_ma77d.h.

6.95.2.30 int ma77_info_d::unused

Definition at line 128 of file hsl_ma77d.h.

6.95.2.31 ma77pkgtype_d_ma77_info_d::usmall

Definition at line 129 of file hsl_ma77d.h.

6.95.2.32 int ma77_info_d::ispare[5]

Definition at line 134 of file hsl_ma77d.h.

6.95.2.33 long int ma77_info_d::lspare[5]

Definition at line 134 of file hsl_ma77d.h.

6.95.2.34 ma77pkgtype_d_ma77_info_d::rspace[5]

Definition at line 134 of file hsl_ma77d.h.

The documentation for this struct was generated from the following file:

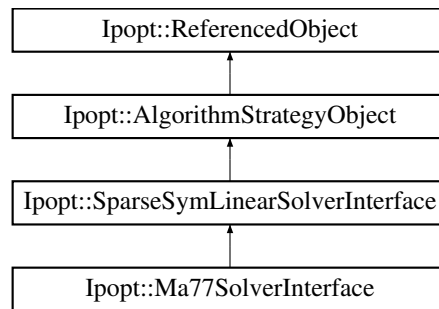
- Algorithm/LinearSolvers/[hsl_ma77d.h](#)

6.96 Ipopt::Ma77SolverInterface Class Reference

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

```
#include <IpMa77SolverInterface.hpp>
```

Inheritance diagram for Ipopt::Ma77SolverInterface:



Public Member Functions

- [Ma77SolverInterface](#) ()
- [~Ma77SolverInterface](#) ()
- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Methods for requesting solution of the linear system.

- [ESymSolverStatus InitializeStructure](#) ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *ia, const [Index](#) *ja)
Method for initializing internal structures.
- double * [GetValuesArrayPtr](#) ()
Method returning an internal array into which the nonzero elements (in the same order as ja) will be stored by the calling routine before a call to MultiSolve with a new_matrix=true (or after a return of MultiSolve with SYMSOLV_CALL_AGA-IN).
- [ESymSolverStatus MultiSolve](#) (bool new_matrix, const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)
Solve operation for multiple right hand sides.
- [Index NumberOfNegEVals](#) () const
Number of negative eigenvalues detected during last factorization.
- bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.

- [EMatrixFormat](#) [MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Methods related to the detection of linearly dependent

rows in a matrix

- bool [ProvidesDegeneracyDetection](#) () const
Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.
- [ESymSolverStatus](#) [DetermineDependentRows](#) (const [Index](#) *ia, const [Index](#) *ja, std::list< [Index](#) > &c_deps)
This method determines the list of row indices of the linearly dependent rows.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)

Private Types

- enum [order_opts](#) { [ORDER_AMD](#), [ORDER_METIS](#) }

Private Attributes

- int [ndim_](#)
- double * [val_](#)
- int [numneg_](#)
- void * [keep_](#)
- bool [pivtol_changed_](#)
- struct [ma77_control](#) [control_](#)
- double [umax_](#)
- int [ordering_](#)

Additional Inherited Members

6.96.1 Detailed Description

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

This defines the general interface to linear solvers for sparse symmetric indefinite matrices. The matrices can be provided either in "triplet format" (like for Harwell's MA27 solver), or in compressed sparse row (CSR) format for the lower triangular part of the symmetric matrix.

The solver should be able to compute the interia of the matrix, or more specifically, the number of negative eigenvalues in the factorized matrix.

This interface is used by the calling objective in the following way:

1. The [InitializeImpl](#) method is called at the very beginning (for every optimization run), which allows the linear solver object to retrieve options given in the [OptionsList](#) (such as pivot tolerances etc). At this point, some internal data can also be initialized.
2. The calling class calls [MatrixFormat](#) to find out which matrix representation the linear solver requires. The possible options are [Triplet_Format](#), as well as [CSR_Format_0_Offset](#) and [CSR_Format_1_Offset](#). The difference between the last two is that for [CSR_Format_0_Offset](#) the counting of the element position in the ia and ja arrays starts are 0 (C-style numbering), whereas for the other one it starts at 1 (Fortran-style numbering).

3. After this, the InitializeStructure method is called (once). Here, the structure of the matrix is provided. If the linear solver requires a symbolic preprocessing phase that can be done without knowledge of the matrix element values, it can be done here.
4. The calling class will request an array for storing the actual values for a matrix using the GetValuesArrayPtr method. This array must be at least as large as the number of nonzeros in the matrix (as given to this class by the InitializeStructure method call). After a call of this method, the calling class will fill this array with the actual values of the matrix.
5. Every time lateron, when actual solves of a linear system is requested, the calling class will call the MultiSolve to request the solve, possibly for multiple right-hand sides. The flag new_matrix then indicates if the values of the matrix have changed and if a factorization is required, or if an old factorization can be used to do the solve.

Note that the GetValuesArrayPtr method will be called before every call of MultiSolve with new_matrix=true, or before a renewed call of MultiSolve if the most previous return value was SYMSOLV_CALL_AGAIN.

1. The calling class might request with NumberOfNegEVals the number of the negative eigenvalues for the original matrix that were detected during the most recently performed factorization.
2. The calling class might ask the linear solver to increase the quality of the solution. For example, if the linear solver uses a pivot tolerance, a larger value should be used for the next solve (which might require a refactorization).
3. Finally, when the destructor is called, the internal storage, also in the linear solver, should be released.

Note, if the matrix is given in triplet format, entries might be listed multiple times, in which case the corresponding elements have to be added.

A note for warm starts: If the option "warm_start_same_structure" is specified with "yes", the algorithm assumes that a problem with the same sparsity structure is solved for a repeated time. In that case, the linear solver might reuse information from the previous optimization. See [Ma27TSolverInterface](#) for an example.

Definition at line 103 of file IpMa77SolverInterface.hpp.

6.96.2 Member Enumeration Documentation

6.96.2.1 enum Ipopt::Ma77SolverInterface::order_opts [private]

Enumerator

ORDER_AMD
ORDER_METIS

Definition at line 106 of file IpMa77SolverInterface.hpp.

6.96.3 Constructor & Destructor Documentation

6.96.3.1 Ipopt::Ma77SolverInterface::Ma77SolverInterface () [inline]

Definition at line 124 of file IpMa77SolverInterface.hpp.

6.96.3.2 Ipopt::Ma77SolverInterface::~~Ma77SolverInterface ()

6.96.4 Member Function Documentation

6.96.4.1 static void Ipopt::Ma77SolverInterface::RegisterOptions (SmartPtr<RegisteredOptions> roptions) [static]

6.96.4.2 `bool lpopt::Ma77SolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix)`
`[virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.96.4.3 `ESymSolverStatus lpopt::Ma77SolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * ia, const Index * ja)` `[virtual]`

Method for initializing internal structures.

Here, `ndim` gives the number of rows and columns of the matrix, `nonzeros` give the number of nonzero elements, and `ia` and `ja` give the positions of the nonzero elements, given in the matrix format determined by `MatrixFormat`.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.96.4.4 `double* lpopt::Ma77SolverInterface::GetValuesArrayPtr ()` `[inline],[virtual]`

Method returning an internal array into which the nonzero elements (in the same order as `ja`) will be stored by the calling routine before a call to `MultiSolve` with a `new_matrix=true` (or after a return of `MultiSolve` with `SYMSOLV_CALL_AGAIN`).

The returned array must have space for at least nonzero elements.

Implements [lpopt::SparseSymLinearSolverInterface](#).

Definition at line 152 of file `lpMa77SolverInterface.hpp`.

6.96.4.5 `ESymSolverStatus lpopt::Ma77SolverInterface::MultiSolve (bool new_matrix, const Index * ia, const Index * ja, Index nrhs, double * rhs_vals, bool check_NegEvals, Index numberOfNegEvals)` `[virtual]`

Solve operation for multiple right hand sides.

Solves the linear system $A * x = b$ with multiple right hand sides, where A is the symmetric indefinite matrix. Here, `ia` and `ja` give the positions of the values (in the required matrix data format). The actual values of the matrix will have been given to this object by copying them into the array provided by `GetValuesArrayPtr`. `ia` and `ja` are identical to the ones given to `InitializeStructure`. The flag `new_matrix` is set to true, if the values of the matrix has changed, and a refactorization is required.

The return code is `SYMSOLV_SUCCESS` if the factorization and solves were successful, `SYMSOLV_SINGULAR` if the linear system is singular, and `SYMSOLV_WRONG_INERTIA` if `check_NegEvals` is true and the number of negative eigenvalues in the matrix does not match `numberOfNegEvals`. If `SYMSOLV_CALL_AGAIN` is returned, then the calling function will request the pointer for the array for storing a again (with `GetValuesPtr`), write the values of the nonzero elements into it, and call this `MultiSolve` method again with the same right-hand sides. (This can be done, for example, if the linear solver realized it does not have sufficient memory and needs to redo the factorization; e.g., for MA27.)

The number of right-hand sides is given by `nrhs`, the values of the right-hand sides are given in `rhs_vals` (one full right-hand side stored immediately after the other), and solutions are to be returned in the same array.

`check_NegEvals` will not be chosen true, if [ProvidesInertia\(\)](#) returns false.

Implements [lpopt::SparseSymLinearSolverInterface](#).

6.96.4.6 `Index lpopt::Ma77SolverInterface::NumberOfNegEvals () const` `[inline],[virtual]`

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see [ProvidesInertia](#)).

Implements [lpopt::SparseSymLinearSolverInterface](#).

Definition at line 203 of file `lpMa77SolverInterface.hpp`.

6.96.4.7 `bool Ipopt::Ma77SolverInterface::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

The calling class asks linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.96.4.8 `bool Ipopt::Ma77SolverInterface::ProvidesInertia () const [inline],[virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 222 of file IpMa77SolverInterface.hpp.

6.96.4.9 `EMatrixFormat Ipopt::Ma77SolverInterface::MatrixFormat () const [inline],[virtual]`

Query of requested matrix type that the linear solver understands.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 230 of file IpMa77SolverInterface.hpp.

6.96.4.10 `bool Ipopt::Ma77SolverInterface::ProvidesDegeneracyDetection () const [inline],[virtual]`

Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 241 of file IpMa77SolverInterface.hpp.

6.96.4.11 `ESymSolverStatus Ipopt::Ma77SolverInterface::DetermineDependentRows (const Index * ia, const Index * ja, std::list< Index > & c_deps) [inline],[virtual]`

This method determines the list of row indices of the linearly dependent rows.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 247 of file IpMa77SolverInterface.hpp.

6.96.5 Member Data Documentation**6.96.5.1** `int Ipopt::Ma77SolverInterface::ndim_ [private]`

Definition at line 111 of file IpMa77SolverInterface.hpp.

6.96.5.2 `double* Ipopt::Ma77SolverInterface::val_ [private]`

Definition at line 112 of file IpMa77SolverInterface.hpp.

6.96.5.3 `int Ipopt::Ma77SolverInterface::numneg_ [private]`

Definition at line 113 of file IpMa77SolverInterface.hpp.

6.96.5.4 `void* Ipopt::Ma77SolverInterface::keep_ [private]`

Definition at line 114 of file IpMa77SolverInterface.hpp.

6.96.5.5 `bool lpopt::Ma77SolverInterface::pivot_changed_ [private]`

Definition at line 115 of file `lpMa77SolverInterface.hpp`.

6.96.5.6 `struct ma77_control lpopt::Ma77SolverInterface::control_ [private]`

Definition at line 118 of file `lpMa77SolverInterface.hpp`.

6.96.5.7 `double lpopt::Ma77SolverInterface::umax_ [private]`

Definition at line 119 of file `lpMa77SolverInterface.hpp`.

6.96.5.8 `int lpopt::Ma77SolverInterface::ordering_ [private]`

Definition at line 120 of file `lpMa77SolverInterface.hpp`.

The documentation for this class was generated from the following file:

- Algorithm/LinearSolvers/[lpMa77SolverInterface.hpp](#)

6.97 `ma86_control_d` Struct Reference

```
#include <hsl_ma86d.h>
```

Public Attributes

- `int f_arrays`
- `int diagnostics_level`
- `int unit_diagnostics`
- `int unit_error`
- `int unit_warning`
- `int nemin`
- `int nb`
- `int action`
- `int nbi`
- `int pool_size`
- `ma86realtype_d_small_`
- `ma86realtype_d_static_`
- `ma86realtype_d_u`
- `ma86realtype_d_u_min`
- `int scaling`

6.97.1 Detailed Description

Definition at line 27 of file `hsl_ma86d.h`.

6.97.2 Member Data Documentation

6.97.2.1 `int ma86_control_d::f_arrays`

Definition at line 31 of file `hsl_ma86d.h`.

6.97.2.2 `int ma86_control_d::diagnostics_level`

Definition at line 35 of file `hsl_ma86d.h`.

6.97.2.3 `int ma86_control_d::unit_diagnostics`

Definition at line 42 of file `hsl_ma86d.h`.

6.97.2.4 `int ma86_control_d::unit_error`

Definition at line 44 of file `hsl_ma86d.h`.

6.97.2.5 `int ma86_control_d::unit_warning`

Definition at line 46 of file `hsl_ma86d.h`.

6.97.2.6 `int ma86_control_d::nemin`

Definition at line 50 of file `hsl_ma86d.h`.

6.97.2.7 `int ma86_control_d::nb`

Definition at line 52 of file `hsl_ma86d.h`.

6.97.2.8 `int ma86_control_d::action`

Definition at line 56 of file `hsl_ma86d.h`.

6.97.2.9 `int ma86_control_d::nbi`

Definition at line 58 of file `hsl_ma86d.h`.

6.97.2.10 `int ma86_control_d::pool_size`

Definition at line 59 of file `hsl_ma86d.h`.

6.97.2.11 `ma86realtype_d ma86_control_d::small_`

Definition at line 60 of file `hsl_ma86d.h`.

6.97.2.12 `ma86realtype_d ma86_control_d::static_`

Definition at line 61 of file `hsl_ma86d.h`.

6.97.2.13 `ma86realtype_d ma86_control_d::u`

Definition at line 62 of file `hsl_ma86d.h`.

6.97.2.14 `ma86realtype_d ma86_control_d::umin`

Definition at line 63 of file `hsl_ma86d.h`.

6.97.2.15 `int ma86_control_d::scaling`

Definition at line 64 of file `hsl_ma86d.h`.

The documentation for this struct was generated from the following file:

- Algorithm/LinearSolvers/[hsl_ma86d.h](#)

6.98 ma86_info_d Struct Reference

```
#include <hsl_ma86d.h>
```

Public Attributes

- [ma86realtype_d_detlog](#)
- int [detsign](#)
- int [flag](#)
- int [matrix_rank](#)
- int [maxdepth](#)
- int [num_delay](#)
- long [num_factor](#)
- long [num_flops](#)
- int [num_neg](#)
- int [num_nodes](#)
- int [num_nothresh](#)
- int [num_perturbed](#)
- int [num_two](#)
- int [pool_size](#)
- int [stat](#)
- [ma86realtype_d_usmall](#)

6.98.1 Detailed Description

Definition at line 70 of file [hsl_ma86d.h](#).

6.98.2 Member Data Documentation

6.98.2.1 [ma86realtype_d_ma86_info_d::detlog](#)

Definition at line 71 of file [hsl_ma86d.h](#).

6.98.2.2 [int ma86_info_d::detsign](#)

Definition at line 72 of file [hsl_ma86d.h](#).

6.98.2.3 [int ma86_info_d::flag](#)

Definition at line 73 of file [hsl_ma86d.h](#).

6.98.2.4 [int ma86_info_d::matrix_rank](#)

Definition at line 74 of file [hsl_ma86d.h](#).

6.98.2.5 [int ma86_info_d::maxdepth](#)

Definition at line 75 of file [hsl_ma86d.h](#).

6.98.2.6 `int ma86_info_d::num_delay`

Definition at line 76 of file `hsl_ma86d.h`.

6.98.2.7 `long ma86_info_d::num_factor`

Definition at line 77 of file `hsl_ma86d.h`.

6.98.2.8 `long ma86_info_d::num_flops`

Definition at line 78 of file `hsl_ma86d.h`.

6.98.2.9 `int ma86_info_d::num_neg`

Definition at line 79 of file `hsl_ma86d.h`.

6.98.2.10 `int ma86_info_d::num_nodes`

Definition at line 80 of file `hsl_ma86d.h`.

6.98.2.11 `int ma86_info_d::num_nothresh`

Definition at line 81 of file `hsl_ma86d.h`.

6.98.2.12 `int ma86_info_d::num_perturbed`

Definition at line 82 of file `hsl_ma86d.h`.

6.98.2.13 `int ma86_info_d::num_two`

Definition at line 83 of file `hsl_ma86d.h`.

6.98.2.14 `int ma86_info_d::pool_size`

Definition at line 84 of file `hsl_ma86d.h`.

6.98.2.15 `int ma86_info_d::stat`

Definition at line 85 of file `hsl_ma86d.h`.

6.98.2.16 `ma86realtype_d ma86_info_d::usmall`

Definition at line 86 of file `hsl_ma86d.h`.

The documentation for this struct was generated from the following file:

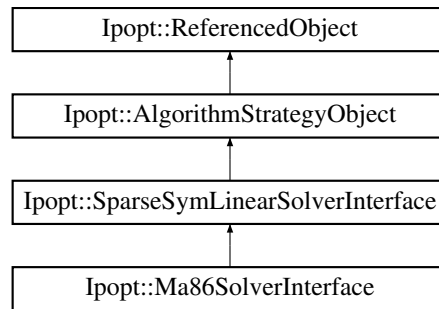
- Algorithm/LinearSolvers/[hsl_ma86d.h](#)

6.99 Ipopt::Ma86SolverInterface Class Reference

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

```
#include <IpMa86SolverInterface.hpp>
```

Inheritance diagram for Ipopt::Ma86SolverInterface:



Public Member Functions

- [Ma86SolverInterface](#) ()
- [~Ma86SolverInterface](#) ()
- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Methods for requesting solution of the linear system.

- [ESymSolverStatus InitializeStructure](#) ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *ia, const [Index](#) *ja)
Method for initializing internal structures.
- double * [GetValuesArrayPtr](#) ()
Method returning an internal array into which the nonzero elements (in the same order as ja) will be stored by the calling routine before a call to MultiSolve with a new_matrix=true (or after a return of MultiSolve with SYMSOLV_CALL_AGAIN).
- [ESymSolverStatus MultiSolve](#) (bool new_matrix, const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)
Solve operation for multiple right hand sides.
- [Index NumberOfNegEVals](#) () const
Number of negative eigenvalues detected during last factorization.
- bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Methods related to the detection of linearly dependent

rows in a matrix

- bool [ProvidesDegeneracyDetection](#) () const
Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.
- [ESymSolverStatus DetermineDependentRows](#) (const [Index](#) *ia, const [Index](#) *ja, std::list< [Index](#) > &c_deps)
This method determines the list of row indices of the linearly dependent rows.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)

Private Types

- enum `order_opts` { `ORDER_AUTO`, `ORDER_AMD`, `ORDER_METIS` }

Private Attributes

- int `ndim_`
- double * `val_`
- int `numneg_`
- `Index` * `order_`
- void * `keep_`
- bool `pivotol_changed_`
- struct `ma86_control` `control_`
- double `umax_`
- int `ordering_`

Additional Inherited Members

6.99.1 Detailed Description

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

This defines the general interface to linear solvers for sparse symmetric indefinite matrices. The matrices can be provided either in "triplet format" (like for Harwell's MA27 solver), or in compressed sparse row (CSR) format for the lower triangular part of the symmetric matrix.

The solver should be able to compute the inertia of the matrix, or more specifically, the number of negative eigenvalues in the factorized matrix.

This interface is used by the calling objective in the following way:

1. The `InitializeImpl` method is called at the very beginning (for every optimization run), which allows the linear solver object to retrieve options given in the `OptionsList` (such as pivot tolerances etc). At this point, some internal data can also be initialized.
2. The calling class calls `MatrixFormat` to find out which matrix representation the linear solver requires. The possible options are `Triplet_Format`, as well as `CSR_Format_0_Offset` and `CSR_Format_1_Offset`. The difference between the last two is that for `CSR_Format_0_Offset` the counting of the element position in the `ia` and `ja` arrays starts are 0 (C-style numbering), whereas for the other one it starts at 1 (Fortran-style numbering).
3. After this, the `InitializeStructure` method is called (once). Here, the structure of the matrix is provided. If the linear solver requires a symbolic preprocessing phase that can be done without knowledge of the matrix element values, it can be done here.
4. The calling class will request an array for storing the actual values for a matrix using the `GetValuesArrayPtr` method. This array must be at least as large as the number of nonzeros in the matrix (as given to this class by the `InitializeStructure` method call). After a call of this method, the calling class will fill this array with the actual values of the matrix.
5. Every time lateron, when actual solves of a linear system is requested, the calling class will call the `MultiSolve` to request the solve, possibly for multiple right-hand sides. The flag `new_matrix` then indicates if the values of the matrix have changed and if a factorization is required, or if an old factorization can be used to do the solve.

Note that the `GetValuesArrayPtr` method will be called before every call of `MultiSolve` with `new_matrix=true`, or before a renewed call of `MultiSolve` if the most previous return value was `SYMSOLV_CALL_AGAIN`.

1. The calling class might request with `NumberOfNegEVals` the number of the negative eigenvalues for the original matrix that were detected during the most recently performed factorization.
2. The calling class might ask the linear solver to increase the quality of the solution. For example, if the linear solver uses a pivot tolerance, a larger value should be used for the next solve (which might require a refactorization).
3. Finally, when the destructor is called, the internal storage, also in the linear solver, should be released.

Note, if the matrix is given in triplet format, entries might be listed multiple times, in which case the corresponding elements have to be added.

A note for warm starts: If the option "warm_start_same_structure" is specified with "yes", the algorithm assumes that a problem with the same sparsity structure is solved for a repeated time. In that case, the linear solver might reuse information from the previous optimization. See [Ma27TSolverInterface](#) for an example.

Definition at line 104 of file `IpMa86SolverInterface.hpp`.

6.99.2 Member Enumeration Documentation

6.99.2.1 `enum Ipopt::Ma86SolverInterface::order_opts` `[private]`

Enumerator

`ORDER_AUTO`
`ORDER_AMD`
`ORDER_METIS`

Definition at line 107 of file `IpMa86SolverInterface.hpp`.

6.99.3 Constructor & Destructor Documentation

6.99.3.1 `Ipopt::Ma86SolverInterface::Ma86SolverInterface ()` `[inline]`

Definition at line 127 of file `IpMa86SolverInterface.hpp`.

6.99.3.2 `Ipopt::Ma86SolverInterface::~~Ma86SolverInterface ()`

6.99.4 Member Function Documentation

6.99.4.1 `static void Ipopt::Ma86SolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > roptions)` `[static]`

6.99.4.2 `bool Ipopt::Ma86SolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix)` `[virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.99.4.3 `ESymSolverStatus Ipopt::Ma86SolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * ia, const Index * ja)` `[virtual]`

Method for initializing internal structures.

Here, `ndim` gives the number of rows and columns of the matrix, `nonzeros` give the number of nonzero elements, and `ia` and `ja` give the positions of the nonzero elements, given in the matrix format determined by `MatrixFormat`.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.99.4.4 `double* Ipopt::Ma86SolverInterface::GetValuesArrayPtr () [inline], [virtual]`

Method returning an internal array into which the nonzero elements (in the same order as *ja*) will be stored by the calling routine before a call to `MultiSolve` with a new `_matrix=true` (or after a return of `MultiSolve` with `SYMSOLV_CALL_AGAIN`).

The returned array must have space for at least nonzero elements.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 155 of file `IpMa86SolverInterface.hpp`.

6.99.4.5 `ESymSolverStatus Ipopt::Ma86SolverInterface::MultiSolve (bool new_matrix, const Index * ia, const Index * ja, Index nrhs, double * rhs_vals, bool check_NegEVals, Index numberOfNegEVals) [virtual]`

Solve operation for multiple right hand sides.

Solves the linear system $A * x = b$ with multiple right hand sides, where A is the symmetric indefinite matrix. Here, *ia* and *ja* give the positions of the values (in the required matrix data format). The actual values of the matrix will have been given to this object by copying them into the array provided by `GetValuesArrayPtr`. *ia* and *ja* are identical to the ones given to `InitializeStructure`. The flag `new_matrix` is set to true, if the values of the matrix has changed, and a refactorization is required.

The return code is `SYMSOLV_SUCCESS` if the factorization and solves were successful, `SYMSOLV_SINGULAR` if the linear system is singular, and `SYMSOLV_WRONG_INERTIA` if `check_NegEVals` is true and the number of negative eigenvalues in the matrix does not match `numberOfNegEVals`. If `SYMSOLV_CALL_AGAIN` is returned, then the calling function will request the pointer for the array for storing a again (with `GetValuesPtr`), write the values of the nonzero elements into it, and call this `MultiSolve` method again with the same right-hand sides. (This can be done, for example, if the linear solver realized it does not have sufficient memory and needs to redo the factorization; e.g., for MA27.)

The number of right-hand sides is given by *nrhs*, the values of the right-hand sides are given in *rhs_vals* (one full right-hand side stored immediately after the other), and solutions are to be returned in the same array.

`check_NegEVals` will not be chosen true, if [ProvidesInertia\(\)](#) returns false.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.99.4.6 `Index Ipopt::Ma86SolverInterface::NumberOfNegEVals () const [inline], [virtual]`

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see [ProvidesInertia](#)).

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 206 of file `IpMa86SolverInterface.hpp`.

6.99.4.7 `bool Ipopt::Ma86SolverInterface::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

The calling class asks linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.99.4.8 `bool Ipopt::Ma86SolverInterface::ProvidesInertia () const [inline], [virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 225 of file IpMa86SolverInterface.hpp.

6.99.4.9 EMatrixFormat Ipopt::Ma86SolverInterface::MatrixFormat () const [inline],[virtual]

Query of requested matrix type that the linear solver understands.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 233 of file IpMa86SolverInterface.hpp.

6.99.4.10 bool Ipopt::Ma86SolverInterface::ProvidesDegeneracyDetection () const [inline],[virtual]

Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 244 of file IpMa86SolverInterface.hpp.

6.99.4.11 ESymSolverStatus Ipopt::Ma86SolverInterface::DetermineDependentRows (const Index * *ia*, const Index * *ja*, std::list< Index > & *c_deps*) [inline],[virtual]

This method determines the list of row indices of the linearly dependent rows.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 250 of file IpMa86SolverInterface.hpp.

6.99.5 Member Data Documentation

6.99.5.1 int Ipopt::Ma86SolverInterface::ndim_ [private]

Definition at line 113 of file IpMa86SolverInterface.hpp.

6.99.5.2 double* Ipopt::Ma86SolverInterface::val_ [private]

Definition at line 114 of file IpMa86SolverInterface.hpp.

6.99.5.3 int Ipopt::Ma86SolverInterface::numneg_ [private]

Definition at line 115 of file IpMa86SolverInterface.hpp.

6.99.5.4 Index* Ipopt::Ma86SolverInterface::order_ [private]

Definition at line 116 of file IpMa86SolverInterface.hpp.

6.99.5.5 void* Ipopt::Ma86SolverInterface::keep_ [private]

Definition at line 117 of file IpMa86SolverInterface.hpp.

6.99.5.6 bool Ipopt::Ma86SolverInterface::pivotl_changed_ [private]

Definition at line 118 of file IpMa86SolverInterface.hpp.

6.99.5.7 struct ma86_control Ipopt::Ma86SolverInterface::control_ [private]

Definition at line 121 of file IpMa86SolverInterface.hpp.

6.99.5.8 double `lpopt::Ma86SolverInterface::umax_` [private]

Definition at line 122 of file `lpMa86SolverInterface.hpp`.

6.99.5.9 int `lpopt::Ma86SolverInterface::ordering_` [private]

Definition at line 123 of file `lpMa86SolverInterface.hpp`.

The documentation for this class was generated from the following file:

- Algorithm/LinearSolvers/[lpMa86SolverInterface.hpp](#)

6.100 ma97_control_d Struct Reference

```
#include <hsl_ma97d.h>
```

Public Attributes

- int [f_arrays](#)
- int [action](#)
- int [nemin](#)
- [ma97realtype_d_](#) multiplier
- int [ordering](#)
- int [print_level](#)
- int [scaling](#)
- [ma97realtype_d_](#) small
- [ma97realtype_d_](#) u
- int [unit_diagnostics](#)
- int [unit_error](#)
- int [unit_warning](#)
- long [factor_min](#)
- int [solve_blas3](#)
- long [solve_min](#)
- int [solve_mf](#)
- [ma97realtype_d_](#) consist_tol
- int [ispare](#) [5]
- [ma97realtype_d_](#) rspar [10]

6.100.1 Detailed Description

Definition at line 35 of file `hsl_ma97d.h`.

6.100.2 Member Data Documentation

6.100.2.1 int `ma97_control_d::f_arrays`

Definition at line 36 of file `hsl_ma97d.h`.

6.100.2.2 int `ma97_control_d::action`

Definition at line 37 of file `hsl_ma97d.h`.

6.100.2.3 int ma97_control_d::nemin

Definition at line 39 of file hsl_ma97d.h.

6.100.2.4 ma97realtype_d_ma97_control_d::multiplier

Definition at line 41 of file hsl_ma97d.h.

6.100.2.5 int ma97_control_d::ordering

Definition at line 42 of file hsl_ma97d.h.

6.100.2.6 int ma97_control_d::print_level

Definition at line 49 of file hsl_ma97d.h.

6.100.2.7 int ma97_control_d::scaling

Definition at line 50 of file hsl_ma97d.h.

6.100.2.8 ma97realtype_d_ma97_control_d::small

Definition at line 51 of file hsl_ma97d.h.

6.100.2.9 ma97realtype_d_ma97_control_d::u

Definition at line 52 of file hsl_ma97d.h.

6.100.2.10 int ma97_control_d::unit_diagnostics

Definition at line 53 of file hsl_ma97d.h.

6.100.2.11 int ma97_control_d::unit_error

Definition at line 54 of file hsl_ma97d.h.

6.100.2.12 int ma97_control_d::unit_warning

Definition at line 55 of file hsl_ma97d.h.

6.100.2.13 long ma97_control_d::factor_min

Definition at line 56 of file hsl_ma97d.h.

6.100.2.14 int ma97_control_d::solve_blas3

Definition at line 57 of file hsl_ma97d.h.

6.100.2.15 long ma97_control_d::solve_min

Definition at line 58 of file hsl_ma97d.h.

6.100.2.16 int ma97_control_d::solve_mf

Definition at line 59 of file hsl_ma97d.h.

6.100.2.17 ma97realtype_d_ma97_control_d::consist_tol

Definition at line 60 of file hsl_ma97d.h.

6.100.2.18 int ma97_control_d::ispare[5]

Definition at line 63 of file hsl_ma97d.h.

6.100.2.19 ma97realtype_d_ma97_control_d::rspace[10]

Definition at line 63 of file hsl_ma97d.h.

The documentation for this struct was generated from the following file:

- Algorithm/LinearSolvers/[hsl_ma97d.h](#)

6.101 ma97_info Struct Reference

```
#include <hsl_ma97d.h>
```

Public Attributes

- int [flag](#)
- int [flag68](#)
- int [flag77](#)
- int [matrix_dup](#)
- int [matrix_rank](#)
- int [matrix_outrange](#)
- int [matrix_missing_diag](#)
- int [maxdepth](#)
- int [maxfront](#)
- int [num_delay](#)
- long [num_factor](#)
- long [num_flops](#)
- int [num_neg](#)
- int [num_sup](#)
- int [num_two](#)
- int [ordering](#)
- int [stat](#)
- int [ispare](#) [5]
- [ma97realtype_d_rspace](#) [10]

6.101.1 Detailed Description

Definition at line 66 of file hsl_ma97d.h.

6.101.2 Member Data Documentation**6.101.2.1 int ma97_info::flag**

Definition at line 67 of file hsl_ma97d.h.

6.101.2.2 int ma97_info::flag68

Definition at line 68 of file hsl_ma97d.h.

6.101.2.3 int ma97_info::flag77

Definition at line 69 of file hsl_ma97d.h.

6.101.2.4 int ma97_info::matrix_dup

Definition at line 70 of file hsl_ma97d.h.

6.101.2.5 int ma97_info::matrix_rank

Definition at line 71 of file hsl_ma97d.h.

6.101.2.6 int ma97_info::matrix_outrange

Definition at line 72 of file hsl_ma97d.h.

6.101.2.7 int ma97_info::matrix_missing_diag

Definition at line 73 of file hsl_ma97d.h.

6.101.2.8 int ma97_info::maxdepth

Definition at line 74 of file hsl_ma97d.h.

6.101.2.9 int ma97_info::maxfront

Definition at line 75 of file hsl_ma97d.h.

6.101.2.10 int ma97_info::num_delay

Definition at line 76 of file hsl_ma97d.h.

6.101.2.11 long ma97_info::num_factor

Definition at line 77 of file hsl_ma97d.h.

6.101.2.12 long ma97_info::num_flops

Definition at line 78 of file hsl_ma97d.h.

6.101.2.13 int ma97_info::num_neg

Definition at line 79 of file hsl_ma97d.h.

6.101.2.14 int ma97_info::num_sup

Definition at line 80 of file hsl_ma97d.h.

6.101.2.15 int ma97_info::num_two

Definition at line 81 of file hsl_ma97d.h.

6.101.2.16 int ma97_info::ordering

Definition at line 82 of file hsl_ma97d.h.

6.101.2.17 int ma97_info::stat

Definition at line 83 of file hsl_ma97d.h.

6.101.2.18 int ma97_info::ispare[5]

Definition at line 86 of file hsl_ma97d.h.

6.101.2.19 ma97realtype_d ma97_info::rspace[10]

Definition at line 86 of file hsl_ma97d.h.

The documentation for this struct was generated from the following file:

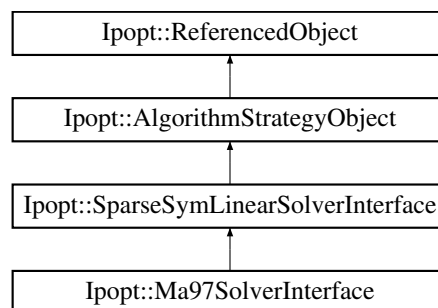
- Algorithm/LinearSolvers/hsl_ma97d.h

6.102 Ipopt::Ma97SolverInterface Class Reference

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

```
#include <IpMa97SolverInterface.hpp>
```

Inheritance diagram for Ipopt::Ma97SolverInterface:



Public Member Functions

- [Ma97SolverInterface](#) ()
- [~Ma97SolverInterface](#) ()
- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Methods for requesting solution of the linear system.

- [ESymSolverStatus](#) [InitializeStructure](#) ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *ia, const [Index](#) *ja)
Method for initializing internal structures.
- double * [GetValuesArrayPtr](#) ()
Method returning an internal array into which the nonzero elements (in the same order as ja) will be stored by the calling routine before a call to MultiSolve with a new_matrix=true (or after a return of MultiSolve with SYMSOLV_CALL_AGA-IN).

- [ESymSolverStatus MultiSolve](#) (bool new_matrix, const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)
Solve operation for multiple right hand sides.
- [Index NumberOfNegEVals](#) () const
Number of negative eigenvalues detected during last factorization.
- bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)

Private Types

- enum [order_opts](#) {
 [ORDER_AUTO](#), [ORDER_BEST](#), [ORDER_AMD](#), [ORDER_METIS](#),
 [ORDER_MATCHED_AUTO](#), [ORDER_MATCHED_AMD](#), [ORDER_MATCHED_METIS](#) }
- enum [scale_opts](#) {
 [SWITCH_NEVER](#), [SWITCH_AT_START](#), [SWITCH_AT_START_REUSE](#), [SWITCH_ON_DEMAND](#),
 [SWITCH_ON_DEMAND_REUSE](#), [SWITCH_NDELAY](#), [SWITCH_NDELAY_REUSE](#), [SWITCH_OD_ND](#),
 [SWITCH_OD_ND_REUSE](#) }

Private Attributes

- int [ndim_](#)
- double * [val_](#)
- int [numneg_](#)
- int [numdelay_](#)
- void * [akeep_](#)
- void * [fkeep_](#)
- bool [pivotl_changed_](#)
- bool [rescale_](#)
- double * [scaling_](#)
- int [fctidx_](#)
- struct [ma97_control](#) [control_](#)
- double [umax_](#)
- int [ordering_](#)
- int [scaling_type_](#)
- enum [scale_opts](#) [switch_](#) [3]
- int [scaling_val_](#) [3]
- int [current_level_](#)
- bool [dump_](#)

Methods related to the detection of linearly dependent

rows in a matrix

- bool [ProvidesDegeneracyDetection](#) () const
Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.
- [ESymSolverStatus DetermineDependentRows](#) (const [Index](#) *ia, const [Index](#) *ja, std::list< [Index](#) > &c_deps)
This method determines the list of row indices of the linearly dependent rows.
- static int [ScaleNameToNum](#) (const std::string &name)
converts a scalign optoin name to its ma97 option number

Additional Inherited Members

6.102.1 Detailed Description

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

This defines the general interface to linear solvers for sparse symmetric indefinite matrices. The matrices can be provided either in "triplet format" (like for Harwell's MA27 solver), or in compressed sparse row (CSR) format for the lower triangular part of the symmetric matrix.

The solver should be able to compute the interia of the matrix, or more specifically, the number of negative eigenvalues in the factorized matrix.

This interface is used by the calling objective in the following way:

1. The InitializeImpl method is called at the very beginning (for every optimization run), which allows the linear solver object to retrieve options given in the [OptionsList](#) (such as pivot tolerances etc). At this point, some internal data can also be initialized.
2. The calling class calls MatrixFormat to find out which matrix representation the linear solver requires. The possible options are Triplet_Format, as well as CSR_Format_0_Offset and CSR_Format_1_Offset. The difference between the last two is that for CSR_Format_0_Offset the counting of the element position in the ia and ja arrays starts are 0 (C-style numbering), whereas for the other one it starts at 1 (Fortran-style numbering).
3. After this, the InitializeStructure method is called (once). Here, the structure of the matrix is provided. If the linear solver requires a symbolic preprocessing phase that can be done without knowledge of the matrix element values, it can be done here.
4. The calling class will request an array for storing the actual values for a matrix using the GetValuesArrayPtr method. This array must be at least as large as the number of nonzeros in the matrix (as given to this class by the InitializeStructure method call). After a call of this method, the calling class will fill this array with the actual values of the matrix.
5. Every time lateron, when actual solves of a linear system is requested, the calling class will call the MultiSolve to request the solve, possibly for mulitple right-hand sides. The flag new_matrix then indicates if the values of the matrix have changed and if a factorization is required, or if an old factorization can be used to do the solve.

Note that the GetValuesArrayPtr method will be called before every call of MultiSolve with new_matrix=true, or before a renewed call of MultiSolve if the most previous return value was SYMSOLV_CALL_AGAIN.

1. The calling class might request with NumberOfNegEVals the number of the negative eigenvalues for the original matrix that were detected during the most recently performed factorization.
2. The calling class might ask the linear solver to increase the quality of the solution. For example, if the linear solver uses a pivot tolerance, a larger value should be used for the next solve (which might require a refactorization).

3. Finally, when the destructor is called, the internal storage, also in the linear solver, should be released.

Note, if the matrix is given in triplet format, entries might be listed multiple times, in which case the corresponding elements have to be added.

A note for warm starts: If the option "warm_start_same_structure" is specified with "yes", the algorithm assumes that a problem with the same sparsity structure is solved for a repeated time. In that case, the linear solver might reuse information from the previous optimization. See [Ma27TSolverInterface](#) for an example.

Definition at line 104 of file IpMa97SolverInterface.hpp.

6.102.2 Member Enumeration Documentation

6.102.2.1 enum Ipopt::Ma97SolverInterface::order_opts [private]

Enumerator

ORDER_AUTO
ORDER_BEST
ORDER_AMD
ORDER_METIS
ORDER_MATCHED_AUTO
ORDER_MATCHED_AMD
ORDER_MATCHED_METIS

Definition at line 107 of file IpMa97SolverInterface.hpp.

6.102.2.2 enum Ipopt::Ma97SolverInterface::scale_opts [private]

Enumerator

SWITCH_NEVER
SWITCH_AT_START
SWITCH_AT_START_REUSE
SWITCH_ON_DEMAND
SWITCH_ON_DEMAND_REUSE
SWITCH_NDELAY
SWITCH_NDELAY_REUSE
SWITCH_OD_ND
SWITCH_OD_ND_REUSE

Definition at line 116 of file IpMa97SolverInterface.hpp.

6.102.3 Constructor & Destructor Documentation

6.102.3.1 Ipopt::Ma97SolverInterface::Ma97SolverInterface () [inline]

Definition at line 151 of file IpMa97SolverInterface.hpp.

6.102.3.2 Ipopt::Ma97SolverInterface::~~Ma97SolverInterface ()

6.102.4 Member Function Documentation

6.102.4.1 `static void Ipopt::Ma97SolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > options)`
[static]

6.102.4.2 `bool Ipopt::Ma97SolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix)`
[virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.102.4.3 `ESymSolverStatus Ipopt::Ma97SolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * ia, const Index * ja)` [virtual]

Method for initializing internal structures.

Here, ndim gives the number of rows and columns of the matrix, nonzeros give the number of nonzero elements, and ia and ja give the positions of the nonzero elements, given in the matrix format determined by MatrixFormat.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.102.4.4 `double* Ipopt::Ma97SolverInterface::GetValuesArrayPtr ()` [inline], [virtual]

Method returning an internal array into which the nonzero elements (in the same order as ja) will be stored by the calling routine before a call to MultiSolve with a new_matrix=true (or after a return of MultiSolve with SYMSOLV_CALL_AGAIN).

The returned array must have space for at least nonzero elements.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 181 of file IpMa97SolverInterface.hpp.

6.102.4.5 `ESymSolverStatus Ipopt::Ma97SolverInterface::MultiSolve (bool new_matrix, const Index * ia, const Index * ja, Index nrhs, double * rhs_vals, bool check_NegEvals, Index numberOfNegEvals)` [virtual]

Solve operation for multiple right hand sides.

Solves the linear system $A * x = b$ with multiple right hand sides, where A is the symmetric indefinite matrix. Here, ia and ja give the positions of the values (in the required matrix data format). The actual values of the matrix will have been given to this object by copying them into the array provided by GetValuesArrayPtr. ia and ja are identical to the ones given to InitializeStructure. The flag new_matrix is set to true, if the values of the matrix has changed, and a refactorization is required.

The return code is SYMSOLV_SUCCESS if the factorization and solves were successful, SYMSOLV_SINGULAR if the linear system is singular, and SYMSOLV_WRONG_INERTIA if check_NegEvals is true and the number of negative eigenvalues in the matrix does not match numberOfNegEvals. If SYMSOLV_CALL_AGAIN is returned, then the calling function will request the pointer for the array for storing a again (with GetValuesPtr), write the values of the nonzero elements into it, and call this MultiSolve method again with the same right-hand sides. (This can be done, for example, if the linear solver realized it does not have sufficient memory and needs to redo the factorization; e.g., for MA27.)

The number of right-hand sides is given by nrhs, the values of the right-hand sides are given in rhs_vals (one full right-hand side stored immediately after the other), and solutions are to be returned in the same array.

check_NegEvals will not be chosen true, if [ProvidesInertia\(\)](#) returns false.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.102.4.6 Index `Ipopt::Ma97SolverInterface::NumberOfNegEVals () const` `[inline],[virtual]`

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see `ProvidesInertia`).

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 232 of file `IpMa97SolverInterface.hpp`.

6.102.4.7 `bool Ipopt::Ma97SolverInterface::IncreaseQuality ()` `[virtual]`

Request to increase quality of solution for next solve.

The calling class asks linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.102.4.8 `bool Ipopt::Ma97SolverInterface::ProvidesInertia () const` `[inline],[virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 251 of file `IpMa97SolverInterface.hpp`.

6.102.4.9 `EMatrixFormat Ipopt::Ma97SolverInterface::MatrixFormat () const` `[inline],[virtual]`

Query of requested matrix type that the linear solver understands.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 259 of file `IpMa97SolverInterface.hpp`.

6.102.4.10 `bool Ipopt::Ma97SolverInterface::ProvidesDegeneracyDetection () const` `[inline],[virtual]`

Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 270 of file `IpMa97SolverInterface.hpp`.

6.102.4.11 `ESymSolverStatus Ipopt::Ma97SolverInterface::DetermineDependentRows (const Index * ia, const Index * ja, std::list< Index > & c_deps)` `[inline],[virtual]`

This method determines the list of row indices of the linearly dependent rows.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 276 of file `IpMa97SolverInterface.hpp`.

6.102.4.12 `static int Ipopt::Ma97SolverInterface::ScaleNameToNum (const std::string & name)` `[static]`

converts a scalign optoin name to its ma97 option number

6.102.5 Member Data Documentation

6.102.5.1 `int Ipopt::Ma97SolverInterface::ndim_ [private]`

Definition at line 128 of file IpMa97SolverInterface.hpp.

6.102.5.2 `double* Ipopt::Ma97SolverInterface::val_ [private]`

Definition at line 129 of file IpMa97SolverInterface.hpp.

6.102.5.3 `int Ipopt::Ma97SolverInterface::numneg_ [private]`

Definition at line 130 of file IpMa97SolverInterface.hpp.

6.102.5.4 `int Ipopt::Ma97SolverInterface::numdelay_ [private]`

Definition at line 131 of file IpMa97SolverInterface.hpp.

6.102.5.5 `void* Ipopt::Ma97SolverInterface::akeep_ [private]`

Definition at line 132 of file IpMa97SolverInterface.hpp.

6.102.5.6 `void* Ipopt::Ma97SolverInterface::fkeep_ [private]`

Definition at line 133 of file IpMa97SolverInterface.hpp.

6.102.5.7 `bool Ipopt::Ma97SolverInterface::pivtol_changed_ [private]`

Definition at line 134 of file IpMa97SolverInterface.hpp.

6.102.5.8 `bool Ipopt::Ma97SolverInterface::rescale_ [private]`

Definition at line 135 of file IpMa97SolverInterface.hpp.

6.102.5.9 `double* Ipopt::Ma97SolverInterface::scaling_ [private]`

Definition at line 136 of file IpMa97SolverInterface.hpp.

6.102.5.10 `int Ipopt::Ma97SolverInterface::fctidx_ [private]`

Definition at line 137 of file IpMa97SolverInterface.hpp.

6.102.5.11 `struct ma97_control Ipopt::Ma97SolverInterface::control_ [private]`

Definition at line 140 of file IpMa97SolverInterface.hpp.

6.102.5.12 `double Ipopt::Ma97SolverInterface::umax_ [private]`

Definition at line 141 of file IpMa97SolverInterface.hpp.

6.102.5.13 `int Ipopt::Ma97SolverInterface::ordering_ [private]`

Definition at line 142 of file IpMa97SolverInterface.hpp.

6.102.5.14 `int Ipopt::Ma97SolverInterface::scaling_type_ [private]`

Definition at line 143 of file IpMa97SolverInterface.hpp.

6.102.5.15 `enum scale_opts Ipopt::Ma97SolverInterface::switch_[3]` [private]

Definition at line 144 of file `IpMa97SolverInterface.hpp`.

6.102.5.16 `int Ipopt::Ma97SolverInterface::scaling_val_[3]` [private]

Definition at line 145 of file `IpMa97SolverInterface.hpp`.

6.102.5.17 `int Ipopt::Ma97SolverInterface::current_level_` [private]

Definition at line 146 of file `IpMa97SolverInterface.hpp`.

6.102.5.18 `bool Ipopt::Ma97SolverInterface::dump_` [private]

Definition at line 147 of file `IpMa97SolverInterface.hpp`.

The documentation for this class was generated from the following file:

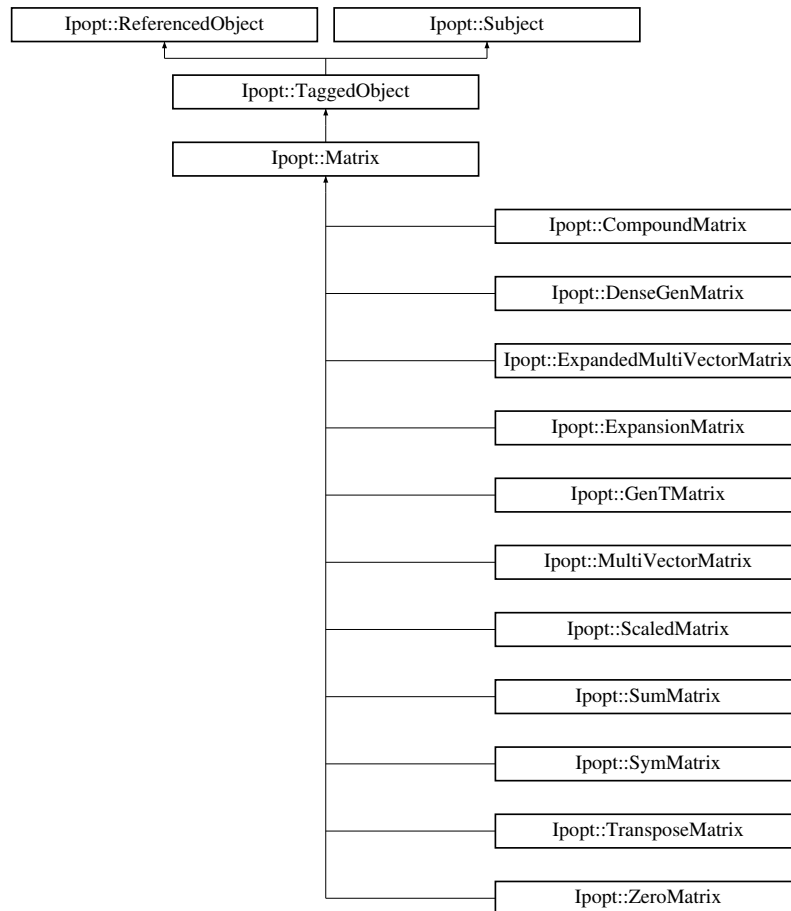
- [Algorithm/LinearSolvers/IpMa97SolverInterface.hpp](#)

6.103 Ipopt::Matrix Class Reference

[Matrix](#) Base Class.

```
#include <IpMatrix.hpp>
```

Inheritance diagram for `Ipopt::Matrix`:



Public Member Functions

- `bool HasValidNumbers () const`
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- `SmartPointer< const MatrixSpace > OwnerSpace () const`
Return the owner *MatrixSpace*.

Constructor/Destructor

- `Matrix (const MatrixSpace *owner_space)`
Constructor.
- `virtual ~Matrix ()`
Destructor.

Operations of the Matrix on a Vector

- `void MultVector (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix-vector multiply.
- `void TransMultVector (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix(transpose) vector multiply.

Methods for specialized operations. A prototype

implementation is provided, but for efficient implementation those should be specially implemented.

- void **AddMSinvZ** (Number alpha, const **Vector** &S, const **Vector** &Z, **Vector** &X) const

$$X = X + \alpha * (\text{Matrix } S^{\{-1\}} Z).$$
- void **SinvBlrmZMTdBr** (Number alpha, const **Vector** &S, const **Vector** &R, const **Vector** &Z, const **Vector** &D, **Vector** &X) const

$$X = S^{\{-1\}} (r + \alpha * Z * M^{\wedge} Td).$$

Information about the size of the matrix

- **Index NRows** () const
Number of rows.
- **Index NCols** () const
Number of columns.

Norms of the individual rows and columns

- void **ComputeRowAMax** (**Vector** &rows_norms, bool init=true) const
Compute the max-norm of the rows in the matrix.
- void **ComputeColAMax** (**Vector** &cols_norms, bool init=true) const
Compute the max-norm of the columns in the matrix.
- virtual void **Print** (**SmartPtr**< const **Journalist** > jnlst, **EJournalLevel** level, **EJournalCategory** category, const std::string &name, **Index** indent=0, const std::string &prefix="") const
Print detailed information about the matrix.
- virtual void **Print** (const **Journalist** &jnlst, **EJournalLevel** level, **EJournalCategory** category, const std::string &name, **Index** indent=0, const std::string &prefix="") const

Protected Member Functions

implementation methods (derived classes MUST

overload these pure virtual protected methods.

- virtual void **MultVectorImpl** (Number alpha, const **Vector** &x, Number beta, **Vector** &y) const =0
Matrix-vector multiply.
- virtual void **TransMultVectorImpl** (Number alpha, const **Vector** &x, Number beta, **Vector** &y) const =0
Matrix(transpose) vector multiply.
- virtual void **AddMSinvZImpl** (Number alpha, const **Vector** &S, const **Vector** &Z, **Vector** &X) const

$$X = X + \alpha * (\text{Matrix } S^{\{-1\}} Z).$$
- virtual void **SinvBlrmZMTdBrlImpl** (Number alpha, const **Vector** &S, const **Vector** &R, const **Vector** &Z, const **Vector** &D, **Vector** &X) const

$$X = S^{\{-1\}} (r + \alpha * Z * M^{\wedge} Td).$$
- virtual bool **HasValidNumbersImpl** () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void **ComputeRowAMaxImpl** (**Vector** &rows_norms, bool init) const =0
Compute the max-norm of the rows in the matrix.
- virtual void **ComputeColAMaxImpl** (**Vector** &cols_norms, bool init) const =0
Compute the max-norm of the columns in the matrix.
- virtual void **PrintImpl** (const **Journalist** &jnlst, **EJournalLevel** level, **EJournalCategory** category, const std::string &name, **Index** indent, const std::string &prefix) const =0
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [Matrix](#) ()
default constructor
- [Matrix](#) (const [Matrix](#) &)
Copy constructor.
- [Matrix](#) & [operator=](#) (const [Matrix](#) &)
Overloaded Equals Operator.

Private Attributes

- const [SmartPtr](#)< const [MatrixSpace](#) > [owner_space_](#)

CachedResults data members

- [TaggedObject::Tag](#) [valid_cache_tag_](#)
- bool [cached_valid_](#)

Additional Inherited Members

6.103.1 Detailed Description

[Matrix](#) Base Class.

This is the base class for all derived matrix types. All Matrices, such as Jacobian and Hessian matrices, as well as possibly the iteration matrices needed for the step computation, are of this type.

Deriving from [Matrix](#): Overload the protected XXX_Impl method.

Definition at line 27 of file IpMatrix.hpp.

6.103.2 Constructor & Destructor Documentation

6.103.2.1 `Ipopt::Matrix::Matrix (const MatrixSpace * owner_space) [inline]`

Constructor.

It has to be given a pointer to the corresponding [MatrixSpace](#).

Definition at line 35 of file IpMatrix.hpp.

6.103.2.2 `virtual Ipopt::Matrix::~Matrix () [inline], [virtual]`

Destructor.

Definition at line 42 of file IpMatrix.hpp.

6.103.2.3 `Ipopt::Matrix::Matrix () [private]`

default constructor

6.103.2.4 Ipopt::Matrix::Matrix (const Matrix &) [private]

Copy constructor.

6.103.3 Member Function Documentation

6.103.3.1 void Ipopt::Matrix::MultVector (Number *alpha*, const Vector & *x*, Number *beta*, Vector & *y*) const [inline]

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$. Do not overload. Overload MultVectorImpl instead.

Definition at line 51 of file IpMatrix.hpp.

6.103.3.2 void Ipopt::Matrix::TransMultVector (Number *alpha*, const Vector & *x*, Number *beta*, Vector & *y*) const [inline]

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$. Do not overload. Overload TransMultVectorImpl instead.

Definition at line 61 of file IpMatrix.hpp.

6.103.3.3 void Ipopt::Matrix::AddMSinvZ (Number *alpha*, const Vector & *S*, const Vector & *Z*, Vector & *X*) const

$X = X + \alpha * (\text{Matrix } S^{-1} Z)$.

Should be implemented efficiently for the ExpansionMatrix

6.103.3.4 void Ipopt::Matrix::SinvBlrmZMTdBr (Number *alpha*, const Vector & *S*, const Vector & *R*, const Vector & *Z*, const Vector & *D*, Vector & *X*) const

$X = S^{-1} (r + \alpha * Z * M^T D)$.

Should be implemented efficiently for the ExpansionMatrix

6.103.3.5 bool Ipopt::Matrix::IsValidNumbers () const

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

6.103.3.6 Index Ipopt::Matrix::NRows () const [inline]

Number of rows.

Definition at line 309 of file IpMatrix.hpp.

6.103.3.7 Index Ipopt::Matrix::NCols () const [inline]

Number of columns.

Definition at line 315 of file IpMatrix.hpp.

6.103.3.8 void Ipopt::Matrix::ComputeRowAMax (Vector & *rows_norms*, bool *init* = true) const [inline]

Compute the max-norm of the rows in the matrix.

The result is stored in *rows_norms*. The vector is assumed to be initialized if *init* is false.

Definition at line 107 of file IpMatrix.hpp.

6.103.3.9 `void Ipopt::Matrix::ComputeColAMax (Vector & cols_norms, bool init = true) const [inline]`

Compute the max-norm of the columns in the matrix.

The result is stored in cols_norms The vector is assumed to be initialized of init is false.

Definition at line 116 of file IpMatrix.hpp.

6.103.3.10 `virtual void Ipopt::Matrix::Print (SmartPtr< const Journalist > jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent = 0, const std::string & prefix = " ") const [virtual]`

Print detailed information about the matrix.

Do not overload. Overload PrintImpl instead.

6.103.3.11 `virtual void Ipopt::Matrix::Print (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent = 0, const std::string & prefix = " ") const [virtual]`

6.103.3.12 `SmartPtr< const MatrixSpace > Ipopt::Matrix::OwnerSpace () const [inline]`

Return the owner [MatrixSpace](#).

Definition at line 321 of file IpMatrix.hpp.

6.103.3.13 `virtual void Ipopt::Matrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [pure virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implemented in [Ipopt::DenseGenMatrix](#), [Ipopt::MultiVectorMatrix](#), [Ipopt::SymTMatrix](#), [Ipopt::DenseSymMatrix](#), [Ipopt::LowRankUpdateSymMatrix](#), [Ipopt::GenTMatrix](#), [Ipopt::CompoundMatrix](#), [Ipopt::CompoundSymMatrix](#), [Ipopt::ExpandedMultiVectorMatrix](#), [Ipopt::ExpansionMatrix](#), [Ipopt::ScaledMatrix](#), [Ipopt::SymScaledMatrix](#), [Ipopt::IdentityMatrix](#), [Ipopt::SumSymMatrix](#), [Ipopt::SumMatrix](#), [Ipopt::DiagMatrix](#), [Ipopt::TransposeMatrix](#), [Ipopt::ZeroMatrix](#), and [Ipopt::ZeroSymMatrix](#).

6.103.3.14 `virtual void Ipopt::Matrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [pure virtual]`

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implemented in [Ipopt::DenseGenMatrix](#), [Ipopt::MultiVectorMatrix](#), [Ipopt::GenTMatrix](#), [Ipopt::CompoundMatrix](#), [Ipopt::ExpandedMultiVectorMatrix](#), [Ipopt::ExpansionMatrix](#), [Ipopt::ScaledMatrix](#), [Ipopt::SymMatrix](#), [Ipopt::SumMatrix](#), [Ipopt::TransposeMatrix](#), [Ipopt::ZeroMatrix](#), and [Ipopt::ZeroSymMatrix](#).

6.103.3.15 `virtual void Ipopt::Matrix::AddMSInvZImpl (Number alpha, const Vector & S, const Vector & Z, Vector & X) const [protected], [virtual]`

$X = X + \alpha * (\text{Matrix} S^{-1}) Z$.

Prototype for this specialize method is provided, but for efficient implementation it should be overloaded for the expansion matrix.

Reimplemented in [Ipopt::CompoundMatrix](#), [Ipopt::ScaledMatrix](#), [Ipopt::ExpansionMatrix](#), and [Ipopt::IdentityMatrix](#).

6.103.3.16 `virtual void Ipopt::Matrix::SinvBlrmZMTdBrlmpl (Number alpha, const Vector & S, const Vector & R, const Vector & Z, const Vector & D, Vector & X) const` [protected],[virtual]

$X = S^{-1} (r + \alpha * Z * M^T d)$.

Should be implemented efficiently for the [ExpansionMatrix](#).

Reimplemented in [Ipopt::CompoundMatrix](#), [Ipopt::ScaledMatrix](#), and [Ipopt::ExpansionMatrix](#).

6.103.3.17 `virtual bool Ipopt::Matrix::IsValidNumbersImpl () const` [inline],[protected],[virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

A default implementation always returning true is provided, but if possible it should be implemented.

Reimplemented in [Ipopt::DenseGenMatrix](#), [Ipopt::MultiVectorMatrix](#), [Ipopt::SymTMatrix](#), [Ipopt::CompoundMatrix](#), [Ipopt::DenseSymMatrix](#), [Ipopt::LowRankUpdateSymMatrix](#), [Ipopt::GenTMatrix](#), [Ipopt::CompoundSymMatrix](#), [Ipopt::ExpandedMultiVectorMatrix](#), [Ipopt::ScaledMatrix](#), [Ipopt::SymScaledMatrix](#), [Ipopt::IdentityMatrix](#), [Ipopt::TransposeMatrix](#), [Ipopt::SumMatrix](#), [Ipopt::SumSymMatrix](#), and [Ipopt::DiagMatrix](#).

Definition at line 178 of file `IpMatrix.hpp`.

6.103.3.18 `virtual void Ipopt::Matrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` [protected],[pure virtual]

Compute the max-norm of the rows in the matrix.

The result is stored in `rows_norms`. The vector is assumed to be initialized.

Implemented in [Ipopt::DenseGenMatrix](#), [Ipopt::MultiVectorMatrix](#), [Ipopt::SymTMatrix](#), [Ipopt::CompoundMatrix](#), [Ipopt::DenseSymMatrix](#), [Ipopt::LowRankUpdateSymMatrix](#), [Ipopt::GenTMatrix](#), [Ipopt::CompoundSymMatrix](#), [Ipopt::ExpandedMultiVectorMatrix](#), [Ipopt::ExpansionMatrix](#), [Ipopt::ScaledMatrix](#), [Ipopt::TransposeMatrix](#), [Ipopt::SymScaledMatrix](#), [Ipopt::IdentityMatrix](#), [Ipopt::SumMatrix](#), [Ipopt::SumSymMatrix](#), [Ipopt::DiagMatrix](#), [Ipopt::ZeroMatrix](#), and [Ipopt::ZeroSymMatrix](#).

6.103.3.19 `virtual void Ipopt::Matrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const` [protected],[pure virtual]

Compute the max-norm of the columns in the matrix.

The result is stored in `cols_norms`. The vector is assumed to be initialized.

Implemented in [Ipopt::DenseGenMatrix](#), [Ipopt::MultiVectorMatrix](#), [Ipopt::CompoundMatrix](#), [Ipopt::LowRankUpdateSymMatrix](#), [Ipopt::DenseSymMatrix](#), [Ipopt::GenTMatrix](#), [Ipopt::ExpandedMultiVectorMatrix](#), [Ipopt::ExpansionMatrix](#), [Ipopt::ScaledMatrix](#), [Ipopt::TransposeMatrix](#), [Ipopt::SymMatrix](#), [Ipopt::SumMatrix](#), [Ipopt::SumSymMatrix](#), [Ipopt::ZeroMatrix](#), and [Ipopt::ZeroSymMatrix](#).

6.103.3.20 `virtual void Ipopt::Matrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected],[pure virtual]

Print detailed information about the matrix.

Implemented in [Ipopt::DenseGenMatrix](#), [Ipopt::MultiVectorMatrix](#), [Ipopt::SymTMatrix](#), [Ipopt::CompoundMatrix](#), [Ipopt::LowRankUpdateSymMatrix](#), [Ipopt::DenseSymMatrix](#), [Ipopt::GenTMatrix](#), [Ipopt::CompoundSymMatrix](#), [Ipopt::ExpandedMultiVectorMatrix](#), [Ipopt::ExpansionMatrix](#), [Ipopt::TransposeMatrix](#), [Ipopt::ScaledMatrix](#), [Ipopt::SymScaledMatrix](#), [Ipopt::IdentityMatrix](#), [Ipopt::SumMatrix](#), [Ipopt::SumSymMatrix](#), [Ipopt::DiagMatrix](#), [Ipopt::ZeroMatrix](#), and [Ipopt::ZeroSymMatrix](#).

6.103.3.21 `Matrix& Ipopt::Matrix::operator= (const Matrix &)` [private]

Overloaded Equals Operator.

6.103.4 Member Data Documentation

6.103.4.1 `const SmartPtr<const MatrixSpace> Ipopt::Matrix::owner_space_` [private]

Definition at line 220 of file IpMatrix.hpp.

6.103.4.2 `TaggedObject::Tag Ipopt::Matrix::valid_cache_tag_` [mutable], [private]

Definition at line 224 of file IpMatrix.hpp.

6.103.4.3 `bool Ipopt::Matrix::cached_valid_` [mutable], [private]

Definition at line 225 of file IpMatrix.hpp.

The documentation for this class was generated from the following file:

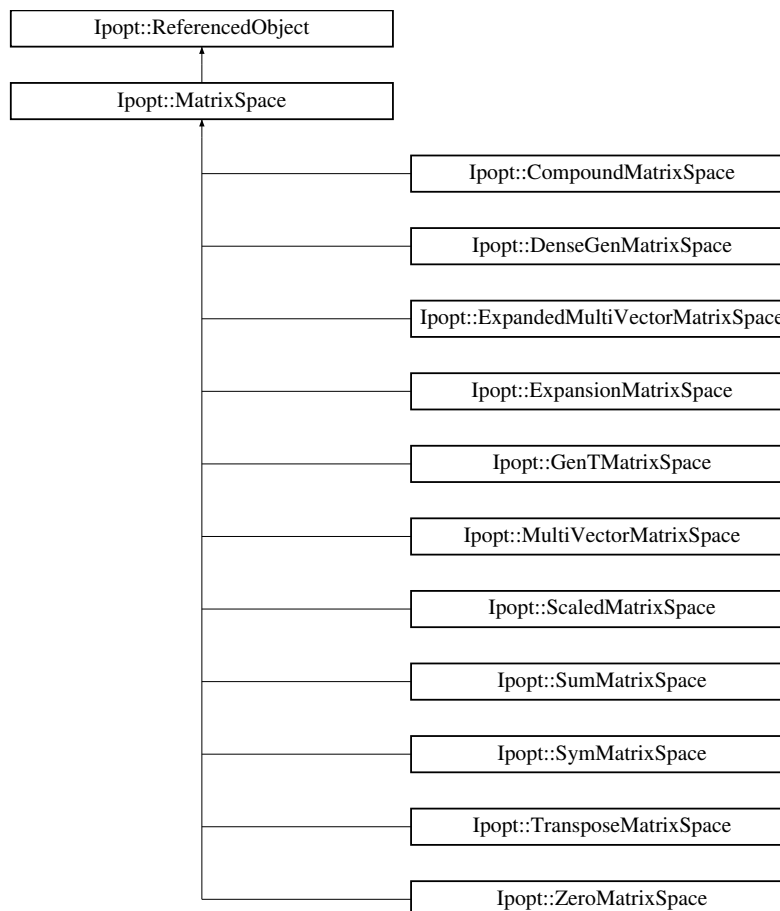
- [LinAlg/IpMatrix.hpp](#)

6.104 Ipopt::MatrixSpace Class Reference

[MatrixSpace](#) base class, corresponding to the [Matrix](#) base class.

```
#include <IpMatrix.hpp>
```

Inheritance diagram for Ipopt::MatrixSpace:



Public Member Functions

- virtual [Matrix](#) * [MakeNew](#) () const =0
Pure virtual method for creating a new [Matrix](#) of the corresponding type.
- [Index NRows](#) () const
Accessor function for the number of rows.
- [Index NCols](#) () const
Accessor function for the number of columns.
- bool [IsMatrixFromSpace](#) (const [Matrix](#) &matrix) const
Method to test if a given matrix belongs to a particular matrix space.

Constructors/Destructors

- [MatrixSpace](#) ([Index](#) nRows, [Index](#) nCols)
Constructor, given the number rows and columns of all matrices generated by this [MatrixSpace](#).
- virtual [~MatrixSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [MatrixSpace](#) ()
default constructor
- [MatrixSpace](#) (const [MatrixSpace](#) &)
Copy constructor.
- [MatrixSpace](#) & [operator=](#) (const [MatrixSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- const [Index](#) [nRows_](#)
Number of rows for all matrices of this type.
- const [Index](#) [nCols_](#)
Number of columns for all matrices of this type.

6.104.1 Detailed Description

[MatrixSpace](#) base class, corresponding to the [Matrix](#) base class.

For each [Matrix](#) implementation, a corresponding [MatrixSpace](#) has to be implemented. A [MatrixSpace](#) is able to create new Matrices of a specific type. The [MatrixSpace](#) should also store information that is common to all Matrices of that type. For example, the dimensions of a [Matrix](#) is stored in the [MatrixSpace](#) base class.

Definition at line 238 of file [lpMatrix.hpp](#).

6.104.2 Constructor & Destructor Documentation

6.104.2.1 Ipopt::MatrixSpace::MatrixSpace (Index *nRows*, Index *nCols*) [inline]

Constructor, given the number rows and columns of all matrices generated by this [MatrixSpace](#).

Definition at line 246 of file IpMatrix.hpp.

6.104.2.2 virtual Ipopt::MatrixSpace::~~MatrixSpace () [inline],[virtual]

Destructor.

Definition at line 253 of file IpMatrix.hpp.

6.104.2.3 Ipopt::MatrixSpace::MatrixSpace () [private]

default constructor

6.104.2.4 Ipopt::MatrixSpace::MatrixSpace (const MatrixSpace &) [private]

Copy constructor.

6.104.3 Member Function Documentation

6.104.3.1 virtual Matrix* Ipopt::MatrixSpace::MakeNew () const [pure virtual]

Pure virtual method for creating a new [Matrix](#) of the corresponding type.

Implemented in [Ipopt::CompoundMatrixSpace](#), [Ipopt::DenseGenMatrixSpace](#), [Ipopt::MultiVectorMatrixSpace](#), [Ipopt::GenTMatrixSpace](#), [Ipopt::ExpansionMatrixSpace](#), [Ipopt::ScaledMatrixSpace](#), [Ipopt::SymScaledMatrixSpace](#), [Ipopt::ExpandedMultiVectorMatrixSpace](#), [Ipopt::SumMatrixSpace](#), [Ipopt::TransposeMatrixSpace](#), [Ipopt::SymMatrixSpace](#), [Ipopt::ZeroMatrixSpace](#), and [Ipopt::ZeroSymMatrixSpace](#).

6.104.3.2 Index Ipopt::MatrixSpace::NRows () const [inline]

Accessor function for the number of rows.

Definition at line 263 of file IpMatrix.hpp.

6.104.3.3 Index Ipopt::MatrixSpace::NCols () const [inline]

Accessor function for the number of columns.

Definition at line 268 of file IpMatrix.hpp.

6.104.3.4 bool Ipopt::MatrixSpace::IsMatrixFromSpace (const Matrix & *matrix*) const [inline]

Method to test if a given matrix belongs to a particular matrix space.

Definition at line 276 of file IpMatrix.hpp.

6.104.3.5 MatrixSpace& Ipopt::MatrixSpace::operator= (const MatrixSpace &) [private]

Overloaded Equals Operator.

6.104.4 Member Data Documentation

6.104.4.1 `const Index Ipopt::MatrixSpace::nRows_ [private]`

Number of rows for all matrices of this type.

Definition at line 301 of file `IpMatrix.hpp`.

6.104.4.2 `const Index Ipopt::MatrixSpace::nCols_ [private]`

Number of columns for all matrices of this type.

Definition at line 303 of file `IpMatrix.hpp`.

The documentation for this class was generated from the following file:

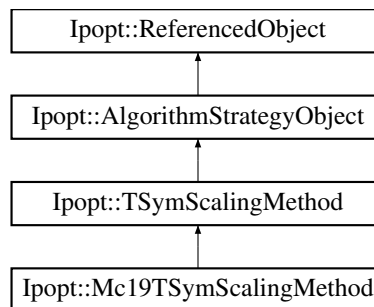
- [LinAlg/IpMatrix.hpp](#)

6.105 `Ipopt::Mc19TSymScalingMethod` Class Reference

Class for the method for computing scaling factors for symmetric matrices in triplet format, using MC19.

```
#include <IpMc19TSymScalingMethod.hpp>
```

Inheritance diagram for `Ipopt::Mc19TSymScalingMethod`:



Public Member Functions

- virtual bool `InitializeImpl` (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool `ComputeSymTScalingFactors` ([Index](#) n, [Index](#) nnz, const [ipfint](#) *airn, const [ipfint](#) *ajcn, const double *a, double *scaling_factors)
Method for computing the symmetric scaling factors, given the symmetric matrix in triplet (MA27) format.

Constructor/Destructor

- [Mc19TSymScalingMethod](#) ()
- virtual `~Mc19TSymScalingMethod` ()

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [Mc19TSymScalingMethod](#) (const [Mc19TSymScalingMethod](#) &)
Copy Constructor.
- void [operator=](#) (const [Mc19TSymScalingMethod](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.105.1 Detailed Description

Class for the method for computing scaling factors for symmetric matrices in triplet format, using MC19.

Definition at line 21 of file `IpMc19TSymScalingMethod.hpp`.

6.105.2 Constructor & Destructor Documentation

6.105.2.1 `Ipopt::Mc19TSymScalingMethod::Mc19TSymScalingMethod () [inline]`

Definition at line 26 of file `IpMc19TSymScalingMethod.hpp`.

6.105.2.2 `virtual Ipopt::Mc19TSymScalingMethod::~~Mc19TSymScalingMethod () [inline],[virtual]`

Definition at line 29 of file `IpMc19TSymScalingMethod.hpp`.

6.105.2.3 `Ipopt::Mc19TSymScalingMethod::Mc19TSymScalingMethod (const Mc19TSymScalingMethod &) [private]`

Copy Constructor.

6.105.3 Member Function Documentation

6.105.3.1 `virtual bool Ipopt::Mc19TSymScalingMethod::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::TSymScalingMethod](#).

6.105.3.2 `virtual bool Ipopt::Mc19TSymScalingMethod::ComputeSymTScalingFactors (Index n, Index nnz, const ipfint * airn, const ipfint * ajcn, const double * a, double * scaling_factors) [virtual]`

Method for computing the symmetric scaling factors, given the symmetric matrix in triplet (MA27) format.

6.105.3.3 `void Ipopt::Mc19TSymScalingMethod::operator= (const Mc19TSymScalingMethod &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

- [Algorithm/LinearSolvers/IpMc19TSymScalingMethod.hpp](#)

6.106 mc68_control Struct Reference

```
#include <hsl_mc68i.h>
```

Public Attributes

- int [f_array_in](#)
- int [f_array_out](#)
- long [min_l_workspace](#)
- int [lp](#)
- int [wp](#)
- int [mp](#)
- int [nemin](#)
- int [print_level](#)
- int [row_full_thresh](#)
- int [row_search](#)

6.106.1 Detailed Description

Definition at line 27 of file `hsl_mc68i.h`.

6.106.2 Member Data Documentation

6.106.2.1 int mc68_control::f_array_in

Definition at line 29 of file `hsl_mc68i.h`.

6.106.2.2 int mc68_control::f_array_out

Definition at line 30 of file `hsl_mc68i.h`.

6.106.2.3 long mc68_control::min_l_workspace

Definition at line 34 of file `hsl_mc68i.h`.

6.106.2.4 int mc68_control::lp

Definition at line 39 of file `hsl_mc68i.h`.

6.106.2.5 int mc68_control::wp

Definition at line 40 of file `hsl_mc68i.h`.

6.106.2.6 int mc68_control::mp

Definition at line 41 of file `hsl_mc68i.h`.

6.106.2.7 int mc68_control::nemin

Definition at line 42 of file `hsl_mc68i.h`.

6.106.2.8 int mc68_control::print_level

Definition at line 43 of file `hsl_mc68i.h`.

6.106.2.9 int mc68_control::row_full_thresh

Definition at line 44 of file `hsl_mc68i.h`.

6.106.2.10 int mc68_control::row_search

Definition at line 45 of file hsl_mc68i.h.

The documentation for this struct was generated from the following file:

- Algorithm/LinearSolvers/[hsl_mc68i.h](#)

6.107 mc68_info Struct Reference

```
#include <hsl_mc68i.h>
```

Public Attributes

- int [flag](#)
- int [iostat](#)
- int [stat](#)
- int [out_range](#)
- int [duplicate](#)
- int [n_compressions](#)
- int [n_zero_eigs](#)
- long [l_workspace](#)
- int [zb01_info](#)
- int [n_dense_rows](#)

6.107.1 Detailed Description

Definition at line 48 of file hsl_mc68i.h.

6.107.2 Member Data Documentation

6.107.2.1 int mc68_info::flag

Definition at line 49 of file hsl_mc68i.h.

6.107.2.2 int mc68_info::iostat

Definition at line 50 of file hsl_mc68i.h.

6.107.2.3 int mc68_info::stat

Definition at line 51 of file hsl_mc68i.h.

6.107.2.4 int mc68_info::out_range

Definition at line 52 of file hsl_mc68i.h.

6.107.2.5 int mc68_info::duplicate

Definition at line 53 of file hsl_mc68i.h.

6.107.2.6 int mc68_info::n_compressions

Definition at line 54 of file hsl_mc68i.h.

6.107.2.7 int mc68_info::n_zero_eigs

Definition at line 55 of file hsl_mc68i.h.

6.107.2.8 long mc68_info::l_workspace

Definition at line 57 of file hsl_mc68i.h.

6.107.2.9 int mc68_info::zb01_info

Definition at line 61 of file hsl_mc68i.h.

6.107.2.10 int mc68_info::n_dense_rows

Definition at line 62 of file hsl_mc68i.h.

The documentation for this struct was generated from the following file:

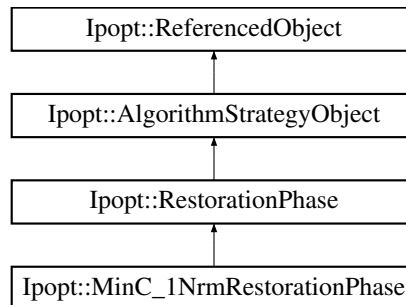
- Algorithm/LinearSolvers/[hsl_mc68i.h](#)

6.108 Ipopt::MinC_1NrmRestorationPhase Class Reference

Restoration Phase that minimizes the 1-norm of the constraint violation - using the interior point method ([Ipopt](#)).

```
#include <IpRestoMinC_1Nrm.hpp>
```

Inheritance diagram for Ipopt::MinC_1NrmRestorationPhase:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
Overloaded from AlgorithmStrategy case class.

Constructors/Destructors

- [MinC_1NrmRestorationPhase](#) ([IpoptAlgorithm](#) &resto_alg, const [SmartPtr](#)< [EqMultiplierCalculator](#) > &eq_mult_calculator)
Constructor, taking strategy objects.
- virtual [~MinC_1NrmRestorationPhase](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Protected Member Functions

- virtual bool [PerformRestoration](#) ()
Overloaded method from [RestorationPhase](#).

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [MinC_1NrmRestorationPhase](#) ()
Default Constructor.
- [MinC_1NrmRestorationPhase](#) (const [MinC_1NrmRestorationPhase](#) &)
Copy Constructor.
- void [operator=](#) (const [MinC_1NrmRestorationPhase](#) &)
Overloaded Equals Operator.

Auxilliary methods

- void [ComputeBoundMultiplierStep](#) ([Vector](#) &delta_z, const [Vector](#) &curr_z, const [Vector](#) &curr_slack, const [Vector](#) &trial_slack)
Method for computing "primal-dual" step in bound multipliers, given step in slacks.

Private Attributes

- [SmartPtr](#)< [OptionsList](#) > resto_options_
Copy of original options, which is required to initialize the [Ipopt](#) algorithm strategy object before restoration phase is started.
- [Index](#) count_restorations_
Counter for the number of time that [PerformRestoration](#) is called.

Strategy objects

- [SmartPtr](#)< [IpoptAlgorithm](#) > resto_alg_
- [SmartPtr](#)< [EqMultiplierCalculator](#) > eq_mult_calculator_

Algorithmic parameters

- [Number](#) constr_mult_reset_threshold_
• [Number](#) bound_mult_reset_threshold_
Maximal allowed value of a bound multiplier after restoration phase.
- bool [expect_infeasible_problem](#)_
Indicates whether problem can be expected to be infeasible.
- [Number](#) constr_viol_tol_
Constraint violation tolerance.
- [Number](#) resto_failure_feasibility_threshold_
Primal infeasibility tolerance for declaring failure of restoration phase when the non-regular termination tests are met.

6.108.1 Detailed Description

Restoration Phase that minimizes the 1-norm of the constraint violation - using the interior point method ([lpopt](#)).

Definition at line 22 of file `IpRestoMinC_1Nrm.hpp`.

6.108.2 Constructor & Destructor Documentation

6.108.2.1 `Ipopt::MinC_1NrmRestorationPhase::MinC_1NrmRestorationPhase (IpoptAlgorithm & resto_alg, const SmartPtr< EqMultiplierCalculator > & eq_mult_calculator)`

Constructor, taking strategy objects.

The `resto_alg` strategy object is the restoration phase [lpopt](#) algorithm. The `eq_mult_calculator` is used to reinitialize the equality constraint multipliers after the restoration phase algorithm has finished - unless it is NULL, in which case the multipliers are set to 0.

6.108.2.2 `virtual Ipopt::MinC_1NrmRestorationPhase::~~MinC_1NrmRestorationPhase () [virtual]`

Default destructor.

6.108.2.3 `Ipopt::MinC_1NrmRestorationPhase::MinC_1NrmRestorationPhase () [private]`

Default Constructor.

6.108.2.4 `Ipopt::MinC_1NrmRestorationPhase::MinC_1NrmRestorationPhase (const MinC_1NrmRestorationPhase &) [private]`

Copy Constructor.

6.108.3 Member Function Documentation

6.108.3.1 `virtual bool Ipopt::MinC_1NrmRestorationPhase::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

Overloaded from AlgorithmStrategy case class.

Implements [Ipopt::RestorationPhase](#).

6.108.3.2 `static void Ipopt::MinC_1NrmRestorationPhase::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for IpoptType.

6.108.3.3 `virtual bool Ipopt::MinC_1NrmRestorationPhase::PerformRestoration () [protected], [virtual]`

Overloaded method from [RestorationPhase](#).

Implements [Ipopt::RestorationPhase](#).

6.108.3.4 `void Ipopt::MinC_1NrmRestorationPhase::operator= (const MinC_1NrmRestorationPhase &) [private]`

Overloaded Equals Operator.

6.108.3.5 void Ipopt::MinC_1NrmRestorationPhase::ComputeBoundMultiplierStep (Vector & *delta_z*, const Vector & *curr_z*, const Vector & *curr_slack*, const Vector & *trial_slack*) [private]

Method for computing "primal-dual" step in bound multipliers, given step in slacks.

6.108.4 Member Data Documentation

6.108.4.1 SmartPtr<IpoptAlgorithm> Ipopt::MinC_1NrmRestorationPhase::resto_alg_ [private]

Definition at line 72 of file IpRestoMinC_1Nrm.hpp.

6.108.4.2 SmartPtr<EqMultiplierCalculator> Ipopt::MinC_1NrmRestorationPhase::eq_mult_calculator_ [private]

Definition at line 73 of file IpRestoMinC_1Nrm.hpp.

6.108.4.3 SmartPtr<OptionsList> Ipopt::MinC_1NrmRestorationPhase::resto_options_ [private]

Copy of original options, which is required to initialize the [Ipopt](#) algorithm strategy object before restoration phase is started.

Definition at line 79 of file IpRestoMinC_1Nrm.hpp.

6.108.4.4 Number Ipopt::MinC_1NrmRestorationPhase::constr_mult_reset_threshold_ [private]

Definition at line 83 of file IpRestoMinC_1Nrm.hpp.

6.108.4.5 Number Ipopt::MinC_1NrmRestorationPhase::bound_mult_reset_threshold_ [private]

Maximal allowed value of a bound multiplier after restoration phase.

Definition at line 87 of file IpRestoMinC_1Nrm.hpp.

6.108.4.6 bool Ipopt::MinC_1NrmRestorationPhase::expect_infeasible_problem_ [private]

Indicates whether problem can be expected to be infeasible.

This will request the to set kappa_resto to a small value for the first time the restoration phase is called. (ToDo)

Definition at line 91 of file IpRestoMinC_1Nrm.hpp.

6.108.4.7 Number Ipopt::MinC_1NrmRestorationPhase::constr_viol_tol_ [private]

Constraint violation tolerance.

Definition at line 93 of file IpRestoMinC_1Nrm.hpp.

6.108.4.8 Number Ipopt::MinC_1NrmRestorationPhase::resto_failure_feasibility_threshold_ [private]

Primal infeasibility tolerance for declaring failure of restoration phase when the non-regular termination tests are met.

Definition at line 97 of file IpRestoMinC_1Nrm.hpp.

6.108.4.9 Index Ipopt::MinC_1NrmRestorationPhase::count_restorations_ [private]

Counter for the number of time that PerformRestoration is called.

Definition at line 102 of file IpRestoMinC_1Nrm.hpp.

The documentation for this class was generated from the following file:

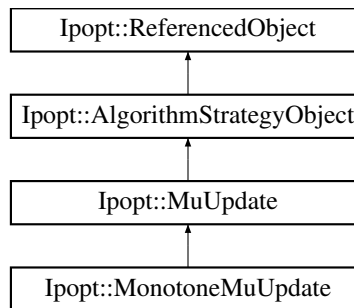
- [Algorithm/IpRestoMinC_1Nrm.hpp](#)

6.109 Ipopt::MonotoneMuUpdate Class Reference

Monotone Mu Update.

```
#include <IpMonotoneMuUpdate.hpp>
```

Inheritance diagram for Ipopt::MonotoneMuUpdate:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
Initialize method - overloaded from [AlgorithmStrategyObject](#).
- virtual bool [UpdateBarrierParameter](#) ()
Method for determining the barrier parameter for the next iteration.

Constructors/Destructors

- [MonotoneMuUpdate](#) (const [SmartPtr](#)< [LineSearch](#) > &linesearch)
Default Constructor.
- virtual [~MonotoneMuUpdate](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)
Methods for IpoptType.

Private Member Functions

- void [CalcNewMuAndTau](#) ([Number](#) &new_mu, [Number](#) &new_tau)
Internal method for computing the new values for mu and tau.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [MonotoneMuUpdate](#) ()
- [MonotoneMuUpdate](#) (const [MonotoneMuUpdate](#) &)
Copy Constructor.
- void [operator=](#) (const [MonotoneMuUpdate](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [LineSearch](#) > [linesearch_](#)
- bool [initialized_](#)
Flag indicating whether the method has been called at least once so far.
- bool [first_iter_resto_](#)
If true, no modification of the barrier parameter will be done at the first call of Update (fix for the restoration phase - we should clean that up!)

Algorithmic parameters

- [Number](#) [mu_init_](#)
Initial value of the barrier parameter.
- [Number](#) [barrier_tol_factor_](#)
- [Number](#) [mu_linear_decrease_factor_](#)
- [Number](#) [mu_superlinear_decrease_power_](#)
- bool [mu_allow_fast_monotone_decrease_](#)
- [Number](#) [tau_min_](#)
Tau_min for fraction to boundary rule.
- [Number](#) [compl_inf_tol_](#)
- [Number](#) [mu_target_](#)

Additional Inherited Members

6.109.1 Detailed Description

Monotone Mu Update.

This class implements the standard monotone mu update approach.

Definition at line 22 of file IpMonotoneMuUpdate.hpp.

6.109.2 Constructor & Destructor Documentation

6.109.2.1 `Ipopt::MonotoneMuUpdate::MonotoneMuUpdate (const SmartPtr< LineSearch > & linesearch)`

Default Constructor.

6.109.2.2 `virtual Ipopt::MonotoneMuUpdate::~~MonotoneMuUpdate ()` `[virtual]`

Default destructor.

6.109.2.3 `Ipopt::MonotoneMuUpdate::MonotoneMuUpdate ()` `[private]`

6.109.2.4 `Ipopt::MonotoneMuUpdate::MonotoneMuUpdate (const MonotoneMuUpdate &)` `[private]`

Copy Constructor.

6.109.3 Member Function Documentation

6.109.3.1 `virtual bool lpopt::MonotoneMuUpdate::Initializelmpl (const OptionsList & options, const std::string & prefix)`
[virtual]

Initialize method - overloaded from [AlgorithmStrategyObject](#).

Implements [lpopt::MuUpdate](#).

6.109.3.2 `virtual bool lpopt::MonotoneMuUpdate::UpdateBarrierParameter ()` [virtual]

Method for determining the barrier parameter for the next iteration.

When the optimality error for the current barrier parameter is less than a tolerance, the barrier parameter is reduced, and the Reset method of the [LineSearch](#) object linesearch is called.

Implements [lpopt::MuUpdate](#).

6.109.3.3 `static void lpopt::MonotoneMuUpdate::RegisterOptions (const SmartPtr< RegisteredOptions > & roptions)`
[static]

Methods for lpoptType.

6.109.3.4 `void lpopt::MonotoneMuUpdate::operator= (const MonotoneMuUpdate &)` [private]

Overloaded Equals Operator.

6.109.3.5 `void lpopt::MonotoneMuUpdate::CalcNewMuAndTau (Number & new_mu, Number & new_tau)` [private]

Internal method for computing the new values for mu and tau.

6.109.4 Member Data Documentation

6.109.4.1 `Number lpopt::MonotoneMuUpdate::mu_init_` [private]

Initial value of the barrier parameter.

Definition at line 75 of file lpMonotoneMuUpdate.hpp.

6.109.4.2 `Number lpopt::MonotoneMuUpdate::barrier_tol_factor_` [private]

Definition at line 76 of file lpMonotoneMuUpdate.hpp.

6.109.4.3 `Number lpopt::MonotoneMuUpdate::mu_linear_decrease_factor_` [private]

Definition at line 77 of file lpMonotoneMuUpdate.hpp.

6.109.4.4 `Number lpopt::MonotoneMuUpdate::mu_superlinear_decrease_power_` [private]

Definition at line 78 of file lpMonotoneMuUpdate.hpp.

6.109.4.5 `bool lpopt::MonotoneMuUpdate::mu_allow_fast_monotone_decrease_` [private]

Definition at line 79 of file lpMonotoneMuUpdate.hpp.

6.109.4.6 `Number lpopt::MonotoneMuUpdate::tau_min_` [private]

Tau_min for fraction to boundary rule.

Definition at line 81 of file IpMonotoneMuUpdate.hpp.

6.109.4.7 **Number** Ipopt::MonotoneMuUpdate::compl_inf_tol_ [private]

Definition at line 82 of file IpMonotoneMuUpdate.hpp.

6.109.4.8 **Number** Ipopt::MonotoneMuUpdate::mu_target_ [private]

Definition at line 83 of file IpMonotoneMuUpdate.hpp.

6.109.4.9 **SmartPointer<LineSearch>** Ipopt::MonotoneMuUpdate::linesearch_ [private]

Definition at line 86 of file IpMonotoneMuUpdate.hpp.

6.109.4.10 **bool** Ipopt::MonotoneMuUpdate::initialized_ [private]

Flag indicating whether the method has been called at least once so far.

Definition at line 90 of file IpMonotoneMuUpdate.hpp.

6.109.4.11 **bool** Ipopt::MonotoneMuUpdate::first_iter_resto_ [private]

If true, no modification of the barrier parameter will be done at the first call of Update (fix for the restoration phase - we should clean that up!)

Definition at line 95 of file IpMonotoneMuUpdate.hpp.

The documentation for this class was generated from the following file:

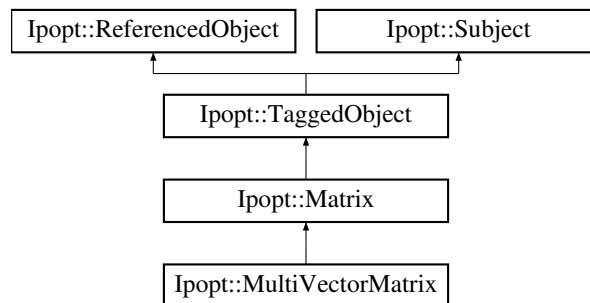
- Algorithm/[IpMonotoneMuUpdate.hpp](#)

6.110 Ipopt::MultiVectorMatrix Class Reference

Class for Matrices with few columns that consists of Vectors.

```
#include <IpMultiVectorMatrix.hpp>
```

Inheritance diagram for Ipopt::MultiVectorMatrix:



Public Member Functions

- **SmartPointer< MultiVectorMatrix >** [MakeNewMultiVectorMatrix](#) () const
Create a new [MultiVectorMatrix](#) from same [MatrixSpace](#).
- **SmartPointer< const Vector >** [GetVector](#) (Index i) const
Get a [Vector](#) in a particular column as a const [Vector](#).

- `SmartPtr< Vector > GetVectorNonConst (Index i)`
Get a `Vector` in a particular column as a non-const `Vector`.
- `void ScaleRows (const Vector &scal_vec)`
Method for scaling the rows of the matrix, using the `ElementWiseMultiply` method for each column vector.
- `void ScaleColumns (const Vector &scal_vec)`
Method for scaling the columns of the matrix, using the `Scal` method for each column vector.
- `void AddOneMultiVectorMatrix (Number a, const MultiVectorMatrix &mv1, Number c)`
Adding another `MultiVectorMatrix`, using the `AddOneVector` methods for the individual column vectors.
- `void AddRightMultMatrix (Number a, const MultiVectorMatrix &U, const Matrix &C, Number b)`
Multiplying a `Matrix` `C` (for now assumed to be a `DenseGenMatrix`) from the right to a `MultiVectorMatrix` `U` and adding the result to this `MultiVectorMatrix` `V`.
- `void FillWithNewVectors ()`
Method for initializing all `Vectors` with new (uninitialized) `Vectors`.
- `void LRMultVector (Number alpha, const Vector &x, Number beta, Vector &y) const`
Method for adding the low-rank update matrix corresponding to this matrix to a vector.
- `SmartPtr< const VectorSpace > ColVectorSpace () const`
`Vector` space for the columns.
- `SmartPtr< const MultiVectorMatrixSpace > MultiVectorMatrixOwnerSpace () const`
Return the `MultiVectorMatrixSpace`.

Constructors / Destructors

- `MultiVectorMatrix (const MultiVectorMatrixSpace *owner_space)`
Constructor, taking the `owner_space`.
- `~MultiVectorMatrix ()`
Destructor.
- `void SetVector (Index i, const Vector &vec)`
Set a particular `Vector` at a given column position, replacing another vector if there has been one.
- `void SetVectorNonConst (Index i, Vector &vec)`

Protected Member Functions

Overloaded methods from Matrix base class

- `virtual void MultVectorImpl (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix-vector multiply.
- `virtual void TransMultVectorImpl (Number alpha, const Vector &x, Number beta, Vector &y) const`
Matrix(transpose) vector multiply.
- `virtual bool HasValidNumbersImpl () const`
Method for determining if all stored numbers are valid (i.e., no `Inf` or `Nan`).
- `virtual void ComputeRowAMaxImpl (Vector &rows_norms, bool init) const`
Compute the max-norm of the rows in the matrix.
- `virtual void ComputeColAMaxImpl (Vector &cols_norms, bool init) const`
Compute the max-norm of the columns in the matrix.
- `virtual void PrintImpl (const Journalist &jnlst, EJournalLevel level, EJournalCategory category, const std::string &name, Index indent, const std::string &prefix) const`
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [MultiVectorMatrix](#) ()
Default Constructor.
- [MultiVectorMatrix](#) (const [MultiVectorMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [MultiVectorMatrix](#) &)
Overloaded Equals Operator.
- const [Vector](#) * [ConstVec](#) ([Index](#) i) const
Method for accessing the internal Vectors internally.
- [Vector](#) * [Vec](#) ([Index](#) i)

Private Attributes

- const [MultiVectorMatrixSpace](#) * [owner_space_](#)
- std::vector< [SmartPtr](#)< const [Vector](#) > > [const_vecs_](#)
space for storing the const [Vector](#)'s
- std::vector< [SmartPtr](#)< [Vector](#) > > [non_const_vecs_](#)
space for storing the non-const [Vector](#)'s

Additional Inherited Members

6.110.1 Detailed Description

Class for Matrices with few columns that consists of Vectors.

Those matrices are for example useful in the implementation of limited memory quasi-Newton methods.

Definition at line 25 of file IpMultiVectorMatrix.hpp.

6.110.2 Constructor & Destructor Documentation

6.110.2.1 Ipopt::MultiVectorMatrix::MultiVectorMatrix (const MultiVectorMatrixSpace * owner_space)

Constructor, taking the owner_space.

6.110.2.2 Ipopt::MultiVectorMatrix::~MultiVectorMatrix () [inline]

Destructor.

Definition at line 225 of file IpMultiVectorMatrix.hpp.

6.110.2.3 Ipopt::MultiVectorMatrix::MultiVectorMatrix () [private]

Default Constructor.

6.110.2.4 `Ipopt::MultiVectorMatrix::MultiVectorMatrix (const MultiVectorMatrix &) [private]`

Copy Constructor.

6.110.3 Member Function Documentation

6.110.3.1 `SmartPointer< MultiVectorMatrix > Ipopt::MultiVectorMatrix::MakeNewMultiVectorMatrix () const [inline]`

Create a new [MultiVectorMatrix](#) from same [MatrixSpace](#).

Definition at line 229 of file `IpMultiVectorMatrix.hpp`.

6.110.3.2 `void Ipopt::MultiVectorMatrix::SetVector (Index i, const Vector & vec)`

Set a particular [Vector](#) at a given column position, replacing another vector if there has been one.

Depending on whether the [Vector](#) is const or not, it is stored in the const or non-const internal column.

6.110.3.3 `void Ipopt::MultiVectorMatrix::SetVectorNonConst (Index i, Vector & vec)`

6.110.3.4 `SmartPointer<const Vector> Ipopt::MultiVectorMatrix::GetVector (Index i) const [inline]`

Get a [Vector](#) in a particular column as a const [Vector](#).

Definition at line 56 of file `IpMultiVectorMatrix.hpp`.

6.110.3.5 `SmartPointer<Vector> Ipopt::MultiVectorMatrix::GetVectorNonConst (Index i) [inline]`

Get a [Vector](#) in a particular column as a non-const [Vector](#).

This is fail if the column has currently only a non-const [Vector](#) stored.

Definition at line 64 of file `IpMultiVectorMatrix.hpp`.

6.110.3.6 `void Ipopt::MultiVectorMatrix::ScaleRows (const Vector & scal_vec)`

Method for scaling the rows of the matrix, using the `ElementWiseMultiply` method for each column vector.

6.110.3.7 `void Ipopt::MultiVectorMatrix::ScaleColumns (const Vector & scal_vec)`

Method for scaling the columns of the matrix, using the `Scal` method for each column vector.

6.110.3.8 `void Ipopt::MultiVectorMatrix::AddOneMultiVectorMatrix (Number a, const MultiVectorMatrix & mv1, Number c)`

Adding another [MultiVectorMatrix](#), using the `AddOneVector` methods for the individual column vectors.

6.110.3.9 `void Ipopt::MultiVectorMatrix::AddRightMultMatrix (Number a, const MultiVectorMatrix & U, const Matrix & C, Number b)`

Multiplying a [Matrix](#) C (for now assumed to be a [DenseGenMatrix](#)) from the right to a [MultiVectorMatrix](#) U and adding the result to this [MultiVectorMatrix](#) V.

$V = a * U * C + b * V.$

6.110.3.10 `void Ipopt::MultiVectorMatrix::FillWithNewVectors ()`

Method for initializing all Vectors with new (uninitialized) Vectors.

6.110.3.11 `void Ipopt::MultiVectorMatrix::LRMultVector (Number alpha, const Vector & x, Number beta, Vector & y) const`

Method for adding the low-rank update matrix corresponding to this matrix to a vector.

If *V* is this [MultiVectorMatrix](#), the operation is $y = \text{beta} * y + \text{alpha} * V * V^T * x$.

6.110.3.12 `SmartPtr< const VectorSpace > Ipopt::MultiVectorMatrix::ColVectorSpace () const` `[inline]`

[Vector](#) space for the columns.

Definition at line 235 of file `IpMultiVectorMatrix.hpp`.

6.110.3.13 `SmartPtr< const MultiVectorMatrixSpace > Ipopt::MultiVectorMatrix::MultiVectorMatrixOwnerSpace () const`
`[inline]`

Return the [MultiVectorMatrixSpace](#).

Definition at line 242 of file `IpMultiVectorMatrix.hpp`.

6.110.3.14 `virtual void Ipopt::MultiVectorMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const` `[protected]`, `[virtual]`

Matrix-vector multiply.

Computes $y = \text{alpha} * \text{Matrix} * x + \text{beta} * y$

Implements [Ipopt::Matrix](#).

6.110.3.15 `virtual void Ipopt::MultiVectorMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const` `[protected]`, `[virtual]`

Matrix(transpose) vector multiply.

Computes $y = \text{alpha} * \text{Matrix}^T * x + \text{beta} * y$

Implements [Ipopt::Matrix](#).

6.110.3.16 `virtual bool Ipopt::MultiVectorMatrix::IsValidNumbersImpl () const` `[protected]`, `[virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.110.3.17 `virtual void Ipopt::MultiVectorMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const`
`[protected]`, `[virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in *rows_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.110.3.18 `virtual void Ipopt::MultiVectorMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const`
`[protected]`, `[virtual]`

Compute the max-norm of the columns in the matrix.

The result is stored in *cols_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.110.3.19 `virtual void Ipopt::MultiVectorMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected], [virtual]

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.110.3.20 `void Ipopt::MultiVectorMatrix::operator= (const MultiVectorMatrix &)` [private]

Overloaded Equals Operator.

6.110.3.21 `const Vector* Ipopt::MultiVectorMatrix::ConstVec (Index i) const` [inline], [private]

Method for accessing the internal Vectors internally.

Definition at line 161 of file `IpMultiVectorMatrix.hpp`.

6.110.3.22 `Vector* Ipopt::MultiVectorMatrix::Vec (Index i)` [inline], [private]

Definition at line 173 of file `IpMultiVectorMatrix.hpp`.

6.110.4 Member Data Documentation

6.110.4.1 `const MultiVectorMatrixSpace* Ipopt::MultiVectorMatrix::owner_space_` [private]

Definition at line 151 of file `IpMultiVectorMatrix.hpp`.

6.110.4.2 `std::vector<SmartPointer<const Vector>> Ipopt::MultiVectorMatrix::const_vecs_` [private]

space for storing the const [Vector](#)'s

Definition at line 154 of file `IpMultiVectorMatrix.hpp`.

6.110.4.3 `std::vector<SmartPointer<Vector>> Ipopt::MultiVectorMatrix::non_const_vecs_` [private]

space for storing the non-const [Vector](#)'s

Definition at line 157 of file `IpMultiVectorMatrix.hpp`.

The documentation for this class was generated from the following file:

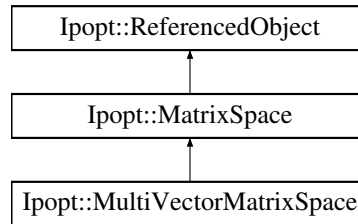
- [LinAlg/IpMultiVectorMatrix.hpp](#)

6.111 Ipopt::MultiVectorMatrixSpace Class Reference

This is the matrix space for [MultiVectorMatrix](#).

```
#include <IpMultiVectorMatrix.hpp>
```

Inheritance diagram for `Ipopt::MultiVectorMatrixSpace`:



Public Member Functions

- [MultiVectorMatrix](#) * [MakeNewMultiVectorMatrix](#) () const
Method for creating a new matrix of this specific type.
- virtual [Matrix](#) * [MakeNew](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.
- [SmartPtr](#)< const [VectorSpace](#) > [ColVectorSpace](#) () const
Accessor method for the [VectorSpace](#) for the columns.

Constructors / Destructors

- [MultiVectorMatrixSpace](#) ([Index](#) ncols, const [VectorSpace](#) &vec_space)
Constructor, given the number of columns (i.e., Vectors to be stored) and given the [VectorSpace](#) for the Vectors.
- [~MultiVectorMatrixSpace](#) ()
Destructor.

Private Attributes

- [SmartPtr](#)< const [VectorSpace](#) > [vec_space_](#)

6.111.1 Detailed Description

This is the matrix space for [MultiVectorMatrix](#).

Definition at line 184 of file `IpMultiVectorMatrix.hpp`.

6.111.2 Constructor & Destructor Documentation

6.111.2.1 `Ipopt::MultiVectorMatrixSpace::MultiVectorMatrixSpace (Index ncols, const VectorSpace & vec_space)`

Constructor, given the number of columns (i.e., Vectors to be stored) and given the [VectorSpace](#) for the Vectors.

6.111.2.2 `Ipopt::MultiVectorMatrixSpace::~~MultiVectorMatrixSpace () [inline]`

Destructor.

Definition at line 196 of file `IpMultiVectorMatrix.hpp`.

6.111.3 Member Function Documentation

6.111.3.1 `MultiVectorMatrix* Ipopt::MultiVectorMatrixSpace::MakeNewMultiVectorMatrix () const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 201 of file IpMultiVectorMatrix.hpp.

6.111.3.2 `virtual Matrix* Ipopt::MultiVectorMatrixSpace::MakeNew () const [inline],[virtual]`

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 208 of file IpMultiVectorMatrix.hpp.

6.111.3.3 `SmartPtr<const VectorSpace> Ipopt::MultiVectorMatrixSpace::ColVectorSpace () const [inline]`

Accessor method for the [VectorSpace](#) for the columns.

Definition at line 214 of file IpMultiVectorMatrix.hpp.

6.111.4 Member Data Documentation

6.111.4.1 `SmartPtr<const VectorSpace> Ipopt::MultiVectorMatrixSpace::vec_space_ [private]`

Definition at line 220 of file IpMultiVectorMatrix.hpp.

The documentation for this class was generated from the following file:

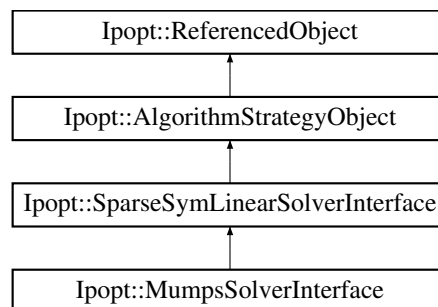
- [LinAlg/IpMultiVectorMatrix.hpp](#)

6.112 Ipopt::MumpsSolverInterface Class Reference

Interface to the linear solver Mumps, derived from [SparseSymLinearSolverInterface](#).

```
#include <IpMumpsSolverInterface.hpp>
```

Inheritance diagram for Ipopt::MumpsSolverInterface:



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [ProvidesDegeneracyDetection](#) () const
Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.
- virtual [ESymSolverStatus](#) [DetermineDependentRows](#) (const [Index](#) *ia, const [Index](#) *ja, std::list< [Index](#) > &c_deps)
This method determines the list of row indices of the linearly dependent rows.

Constructor/Destructor

- [MumpsSolverInterface](#) ()
Constructor.
- virtual [~MumpsSolverInterface](#) ()
Destructor.

Methods for requesting solution of the linear system.

- virtual [ESymSolverStatus InitializeStructure](#) ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *airn, const [Index](#) *ajcn)
Method for initializing internal structures.
- virtual double * [GetValuesArrayPtr](#) ()
Method returning an internal array into which the nonzero elements (in the same order as airn and ajcn) are to be stored by the calling routine before a call to MultiSolve with a new_matrix=true.
- virtual [ESymSolverStatus MultiSolve](#) (bool new_matrix, const [Index](#) *airn, const [Index](#) *ajcn, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)
Solve operation for multiple right hand sides.
- virtual [Index NumberOfNegEVals](#) () const
Number of negative eigenvalues detected during last factorization.
- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions**Default Compiler Generated Methods**

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [MumpsSolverInterface](#) (const [MumpsSolverInterface](#) &)
Copy Constructor.
- void [operator=](#) (const [MumpsSolverInterface](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus SymbolicFactorization](#) ()
Call MUMPS (job=1) to perform symbolic manipulations, and reserve memory.
- [ESymSolverStatus Factorization](#) (bool check_NegEVals, [Index](#) numberOfNegEVals)
Call MUMPS (job=2) to factorize the [Matrix](#).
- [ESymSolverStatus Solve](#) ([Index](#) nrhs, double *rhs_vals)
Call MUMPS (job=3) to do the solve.

Private Attributes

- bool [have_symbolic_factorization_](#)
Flag indicating if symbolic factorization has already been called.

Information about the matrix

- void * [mumps_ptr_](#)
Primary MUMP data structure.

Information about most recent factorization/solve

- [Index negevals_](#)
Number of negative eigenvalues.

Initialization flags

- bool [initialized_](#)
Flag indicating if internal data is initialized.
- bool [pivtol_changed_](#)
Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed.
- bool [refactorize_](#)
Flag that is true if we just requested the values of the matrix again (SYMSOLVER_CALL_AGAIN) and have to factorize again.

Solver specific data/options

- [Number pivtol_](#)
Pivot tolerance.
- [Number pivtolmax_](#)
Maximal pivot tolerance.
- [Index mem_percent_](#)
Percent increase in memory.
- [Index mumps_permuting_scaling_](#)
Permutation and scaling method in MUMPS.
- [Index mumps_pivot_order_](#)
Pivot order in MUMPS.
- [Index mumps_scaling_](#)
Scaling in MUMPS.
- [Number mumps_dep_tol_](#)
Threshold in MUMPS to stay that a constraint is linearly dependent.
- bool [warm_start_same_structure_](#)
*Flag indicating whether the *TNLP* with identical structure has already been solved before.*

Additional Inherited Members

6.112.1 Detailed Description

Interface to the linear solver Mumps, derived from [SparseSymLinearSolverInterface](#).

For details, see description of [SparseSymLinearSolverInterface](#) base class.

Definition at line 26 of file `IpMumpsSolverInterface.hpp`.

6.112.2 Constructor & Destructor Documentation

6.112.2.1 Ipopt::MumpsSolverInterface::MumpsSolverInterface ()

Constructor.

6.112.2.2 virtual Ipopt::MumpsSolverInterface::~~MumpsSolverInterface () [virtual]

Destructor.

6.112.2.3 Ipopt::MumpsSolverInterface::MumpsSolverInterface (const MumpsSolverInterface &) [private]

Copy Constructor.

6.112.3 Member Function Documentation

6.112.3.1 bool Ipopt::MumpsSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.112.3.2 virtual ESymSolverStatus Ipopt::MumpsSolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * airn, const Index * ajcn) [virtual]

Method for initializing internal structures.

Here, ndim gives the number of rows and columns of the matrix, nonzeros give the number of nonzero elements, and airn and ajcn give the positions of the nonzero elements.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.112.3.3 virtual double* Ipopt::MumpsSolverInterface::GetValuesArrayPtr () [virtual]

Method returning an internal array into which the nonzero elements (in the same order as airn and ajcn) are to be stored by the calling routine before a call to MultiSolve with a new_matrix=true.

The returned array must have space for at least nonzero elements.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.112.3.4 virtual ESymSolverStatus Ipopt::MumpsSolverInterface::MultiSolve (bool new_matrix, const Index * aim, const Index * ajcn, Index nrhs, double * rhs_vals, bool check_NegEVals, Index numberOfNegEVals) [virtual]

Solve operation for multiple right hand sides.

Overloaded from [SparseSymLinearSolverInterface](#).

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.112.3.5 virtual Index Ipopt::MumpsSolverInterface::NumberOfNegEVals () const [virtual]

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see ProvidesInertia).

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.112.3.6 `virtual bool Ipopt::MumpsSolverInterface::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

Ask linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.112.3.7 `virtual bool Ipopt::MumpsSolverInterface::ProvidesInertia () const [inline],[virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 92 of file `IpMumpsSolverInterface.hpp`.

6.112.3.8 `EMatrixFormat Ipopt::MumpsSolverInterface::MatrixFormat () const [inline],[virtual]`

Query of requested matrix type that the linear solver understands.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 99 of file `IpMumpsSolverInterface.hpp`.

6.112.3.9 `static void Ipopt::MumpsSolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for `IpoptType`.

6.112.3.10 `virtual bool Ipopt::MumpsSolverInterface::ProvidesDegeneracyDetection () const [virtual]`

Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

6.112.3.11 `virtual ESymSolverStatus Ipopt::MumpsSolverInterface::DetermineDependentRows (const Index * ia, const Index * ja, std::list< Index > & c_deps) [virtual]`

This method determines the list of row indices of the linearly dependent rows.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

6.112.3.12 `void Ipopt::MumpsSolverInterface::operator= (const MumpsSolverInterface &) [private]`

Overloaded Equals Operator.

6.112.3.13 `ESymSolverStatus Ipopt::MumpsSolverInterface::SymbolicFactorization () [private]`

Call MUMPS (job=1) to perform symbolic manipulations, and reserve memory.

6.112.3.14 `ESymSolverStatus Ipopt::MumpsSolverInterface::Factorization (bool check_NegEVals, Index numberOfNegEVals) [private]`

Call MUMPS (job=2) to factorize the [Matrix](#).

It is assumed that the first `nonzeros_` element of `a_` contain the values of the matrix to be factorized.

6.112.3.15 ESymSolverStatus Ipopt::MumpsSolverInterface::Solve (Index *nrhs*, double * *rhs_vals*) [private]

Call MUMPS (job=3) to do the solve.

6.112.4 Member Data Documentation**6.112.4.1 void* Ipopt::MumpsSolverInterface::mumps_ptr_ [private]**

Primary MUMP data structure.

Definition at line 139 of file IpMumpsSolverInterface.hpp.

6.112.4.2 Index Ipopt::MumpsSolverInterface::negevals_ [private]

Number of negative eigenvalues.

Definition at line 145 of file IpMumpsSolverInterface.hpp.

6.112.4.3 bool Ipopt::MumpsSolverInterface::initialized_ [private]

Flag indicating if internal data is initialized.

For initialization, this object needs to have seen a matrix

Definition at line 152 of file IpMumpsSolverInterface.hpp.

6.112.4.4 bool Ipopt::MumpsSolverInterface::pivtol_changed_ [private]

Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed.

Definition at line 155 of file IpMumpsSolverInterface.hpp.

6.112.4.5 bool Ipopt::MumpsSolverInterface::refactorize_ [private]

Flag that is true if we just requested the values of the matrix again (SYMSOLVER_CALL_AGAIN) and have to factorize again.

Definition at line 159 of file IpMumpsSolverInterface.hpp.

6.112.4.6 Number Ipopt::MumpsSolverInterface::pivtol_ [private]

Pivot tolerance.

Definition at line 165 of file IpMumpsSolverInterface.hpp.

6.112.4.7 Number Ipopt::MumpsSolverInterface::pivtolmax_ [private]

Maximal pivot tolerance.

Definition at line 168 of file IpMumpsSolverInterface.hpp.

6.112.4.8 Index Ipopt::MumpsSolverInterface::mem_percent_ [private]

Percent increase in memory.

Definition at line 171 of file IpMumpsSolverInterface.hpp.

6.112.4.9 Index Ipopt::MumpsSolverInterface::mumps_permuting_scaling_ [private]

Permutation and scaling method in MUMPS.

Definition at line 174 of file IpMumpsSolverInterface.hpp.

6.112.4.10 Index Ipopt::MumpsSolverInterface::mumps_pivot_order_ [private]

Pivot order in MUMPS.

Definition at line 177 of file IpMumpsSolverInterface.hpp.

6.112.4.11 Index Ipopt::MumpsSolverInterface::mumps_scaling_ [private]

Scaling in MUMPS.

Definition at line 180 of file IpMumpsSolverInterface.hpp.

6.112.4.12 Number Ipopt::MumpsSolverInterface::mumps_dep_tol_ [private]

Threshold in MUMPS to stay that a constraint is linearly dependent.

Definition at line 184 of file IpMumpsSolverInterface.hpp.

6.112.4.13 bool Ipopt::MumpsSolverInterface::warm_start_same_structure_ [private]

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 188 of file IpMumpsSolverInterface.hpp.

6.112.4.14 bool Ipopt::MumpsSolverInterface::have_symbolic_factorization_ [private]

Flag indicating if symbolic factorization has already been called.

Definition at line 193 of file IpMumpsSolverInterface.hpp.

The documentation for this class was generated from the following file:

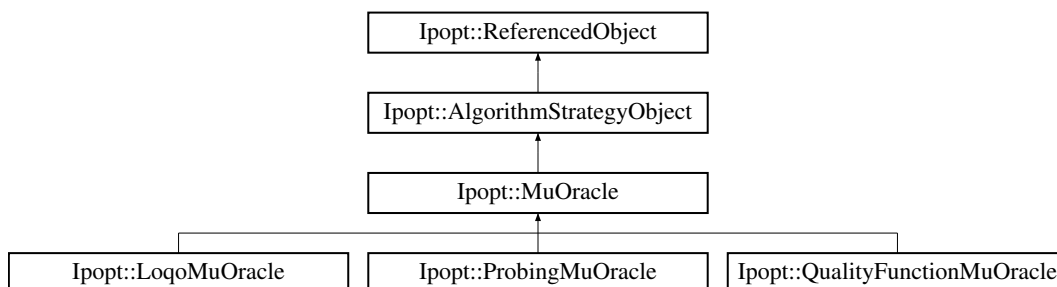
- [Algorithm/LinearSolvers/IpMumpsSolverInterface.hpp](#)

6.113 Ipopt::MuOracle Class Reference

Abstract Base Class for classes that are able to compute a suggested value of the barrier parameter that can be used as an oracle in the NonmontoneMuUpdate class.

```
#include <IpMuOracle.hpp>
```

Inheritance diagram for Ipopt::MuOracle:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0

Initialize method - overloaded from [AlgorithmStrategyObject](#).

- virtual bool [CalculateMu](#) (Number mu_min, Number mu_max, Number &new_mu)=0

Method for computing the value of the barrier parameter that could be used in the current iteration.

Constructors/Destructors

- [MuOracle](#) ()

Default Constructor.

- virtual [~MuOracle](#) ()

Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [MuOracle](#) (const [MuOracle](#) &)

Copy Constructor.

- void [operator=](#) (const [MuOracle](#) &)

Overloaded Equals Operator.

Additional Inherited Members

6.113.1 Detailed Description

Abstract Base Class for classes that are able to compute a suggested value of the barrier parameter that can be used as an oracle in the NonmontoneMuUpdate class.

Definition at line 21 of file IpMuOracle.hpp.

6.113.2 Constructor & Destructor Documentation

6.113.2.1 Ipopt::MuOracle::MuOracle () [inline]

Default Constructor.

Definition at line 27 of file IpMuOracle.hpp.

6.113.2.2 virtual Ipopt::MuOracle::~MuOracle () [inline], [virtual]

Default destructor.

Definition at line 30 of file IpMuOracle.hpp.

6.113.2.3 Ipopt::MuOracle::MuOracle (const MuOracle &) [private]

Copy Constructor.

6.113.3 Member Function Documentation

6.113.3.1 `virtual bool Ipopt::MuOracle::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]`

Initialize method - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::QualityFunctionMuOracle](#), [Ipopt::ProbingMuOracle](#), and [Ipopt::LoqoMuOracle](#).

6.113.3.2 `virtual bool Ipopt::MuOracle::CalculateMu (Number mu_min, Number mu_max, Number & new_mu) [pure virtual]`

Method for computing the value of the barrier parameter that could be used in the current iteration.

Here, mu_min and mu_max are the lower and upper bounds on acceptable values for the barrier parameter. The new value of mu is returned in new_mu, and the method returns false if a new value could not be determined (e.g., because the linear system could not be solved for a predictor step).

Implemented in [Ipopt::ProbingMuOracle](#), [Ipopt::QualityFunctionMuOracle](#), and [Ipopt::LoqoMuOracle](#).

6.113.3.3 `void Ipopt::MuOracle::operator= (const MuOracle &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

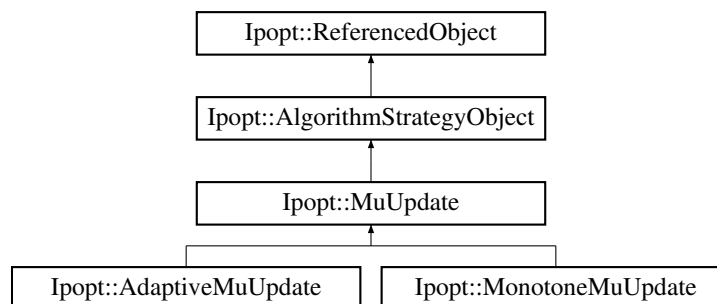
- [Algorithm/IpMuOracle.hpp](#)

6.114 Ipopt::MuUpdate Class Reference

Abstract Base Class for classes that implement methods for computing the barrier and fraction-to-the-boundary rule parameter for the current iteration.

```
#include <IpMuUpdate.hpp>
```

Inheritance diagram for Ipopt::MuUpdate:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
Initialize method - overloaded from [AlgorithmStrategyObject](#).
- virtual bool [UpdateBarrierParameter](#) ()=0
Method for determining the barrier parameter for the next iteration.

Constructors/Destructors

- [MuUpdate](#) ()
Default Constructor.
- virtual [~MuUpdate](#) ()
Default destructor.

Private Member Functions**Default Compiler Generated Methods**

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [MuUpdate](#) (const [MuUpdate](#) &)
Copy Constructor.
- void [operator=](#) (const [MuUpdate](#) &)
Overloaded Equals Operator.

Additional Inherited Members**6.114.1 Detailed Description**

Abstract Base Class for classes that implement methods for computing the barrier and fraction-to-the-boundary rule parameter for the current iteration.

Definition at line 20 of file IpMuUpdate.hpp.

6.114.2 Constructor & Destructor Documentation**6.114.2.1 Ipopt::MuUpdate::MuUpdate () [inline]**

Default Constructor.

Definition at line 26 of file IpMuUpdate.hpp.

6.114.2.2 virtual Ipopt::MuUpdate::~~MuUpdate () [inline],[virtual]

Default destructor.

Definition at line 30 of file IpMuUpdate.hpp.

6.114.2.3 Ipopt::MuUpdate::MuUpdate (const MuUpdate &) [private]

Copy Constructor.

6.114.3 Member Function Documentation**6.114.3.1 virtual bool Ipopt::MuUpdate::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]**

Initialize method - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::AdaptiveMuUpdate](#), and [Ipopt::MonotoneMuUpdate](#).

6.114.3.2 `virtual bool Ipopt::MuUpdate::UpdateBarrierParameter () [pure virtual]`

Method for determining the barrier parameter for the next iteration.

A [LineSearch](#) object is passed, so that this method can call the Reset method in the [LineSearch](#) object, for example when then barrier parameter is changed. This method is also responsible for setting the fraction-to-the-boundary parameter tau. This method returns false if the update could not be performed and the algorithm should revert to an emergency fallback mechanism.

Implemented in [Ipopt::AdaptiveMuUpdate](#), and [Ipopt::MonotoneMuUpdate](#).

6.114.3.3 `void Ipopt::MuUpdate::operator= (const MuUpdate &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

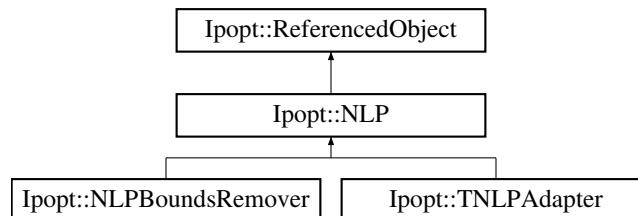
- [Algorithm/lpMuUpdate.hpp](#)

6.115 Ipopt::NLP Class Reference

Brief Class Description.

```
#include <IpNLP.hpp>
```

Inheritance diagram for Ipopt::NLP:



Public Member Functions

- virtual void [GetQuasiNewtonApproximationSpaces](#) (SmartPtr< [VectorSpace](#) > &approx_space, SmartPtr< [Matrix](#) > &P_approx)

Method for obtaining the subspace in which the limited-memory Hessian approximation should be done.

Constructors/Destructors

- [NLP](#) ()
Default constructor.
- virtual [~NLP](#) ()
Default destructor.
- [DECLARE_STD_EXCEPTION](#) (USER_SCALING_NOT_IMPLEMENTED)
Exceptions.
- [DECLARE_STD_EXCEPTION](#) (INVALID_NLP)

NLP Initialization (overload in*derived classes).*

- virtual bool [ProcessOptions](#) (const [OptionsList](#) &options, const std::string &prefix)
Overload if you want the chance to process options or parameters that may be specific to the NLP.
- virtual bool [GetSpaces](#) (SmartPtr< const [VectorSpace](#) > &x_space, SmartPtr< const [VectorSpace](#) > &c_space, SmartPtr< const [VectorSpace](#) > &d_space, SmartPtr< const [VectorSpace](#) > &x_l_space, SmartPtr< const [MatrixSpace](#) > &px_l_space, SmartPtr< const [VectorSpace](#) > &x_u_space, SmartPtr< const [MatrixSpace](#) > &px_u_space, SmartPtr< const [VectorSpace](#) > &d_l_space, SmartPtr< const [MatrixSpace](#) > &pd_l_space, SmartPtr< const [VectorSpace](#) > &d_u_space, SmartPtr< const [MatrixSpace](#) > &pd_u_space, SmartPtr< const [MatrixSpace](#) > &Jac_c_space, SmartPtr< const [MatrixSpace](#) > &Jac_d_space, SmartPtr< const [SymMatrixSpace](#) > &Hess_lagrangian_space)=0
Method for creating the derived vector / matrix types.
- virtual bool [GetBoundsInformation](#) (const [Matrix](#) &Px_L, [Vector](#) &x_L, const [Matrix](#) &Px_U, [Vector](#) &x_U, const [Matrix](#) &Pd_L, [Vector](#) &d_L, const [Matrix](#) &Pd_U, [Vector](#) &d_U)=0
Method for obtaining the bounds information.
- virtual bool [GetStartingPoint](#) (SmartPtr< [Vector](#) > x, bool need_x, SmartPtr< [Vector](#) > y_c, bool need_y_c, SmartPtr< [Vector](#) > y_d, bool need_y_d, SmartPtr< [Vector](#) > z_L, bool need_z_L, SmartPtr< [Vector](#) > z_U, bool need_z_U)=0
Method for obtaining the starting point for all the iterates.
- virtual bool [GetWarmStartIterate](#) ([IteratesVector](#) &warm_start_iterate)
Method for obtaining an entire iterate as a warmstart point.

NLP evaluation routines (overload*in derived classes.*

- virtual bool [Eval_f](#) (const [Vector](#) &x, [Number](#) &f)=0
- virtual bool [Eval_grad_f](#) (const [Vector](#) &x, [Vector](#) &g_f)=0
- virtual bool [Eval_c](#) (const [Vector](#) &x, [Vector](#) &c)=0
- virtual bool [Eval_jac_c](#) (const [Vector](#) &x, [Matrix](#) &jac_c)=0
- virtual bool [Eval_d](#) (const [Vector](#) &x, [Vector](#) &d)=0
- virtual bool [Eval_jac_d](#) (const [Vector](#) &x, [Matrix](#) &jac_d)=0
- virtual bool [Eval_h](#) (const [Vector](#) &x, [Number](#) obj_factor, const [Vector](#) &yc, const [Vector](#) &yd, [SymMatrix](#) &h)=0

NLP solution routines. Have default dummy*implementations that can be overloaded.*

- virtual void [FinalizeSolution](#) ([SolverReturn](#) status, const [Vector](#) &x, const [Vector](#) &z_L, const [Vector](#) &z_U, const [Vector](#) &c, const [Vector](#) &d, const [Vector](#) &y_c, const [Vector](#) &y_d, [Number](#) obj_value, const [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)
This method is called at the very end of the optimization.
- virtual bool [IntermediateCallBack](#) ([AlgorithmMode](#) mode, [Index](#) iter, [Number](#) obj_value, [Number](#) inf_pr, [Number](#) inf_du, [Number](#) mu, [Number](#) d_norm, [Number](#) regularization_size, [Number](#) alpha_du, [Number](#) alpha_pr, [Index](#) ls_trials, const [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)
This method is called once per iteration, after the iteration summary output has been printed.
- virtual void [GetScalingParameters](#) (const SmartPtr< const [VectorSpace](#) > x_space, const SmartPtr< const [VectorSpace](#) > c_space, const SmartPtr< const [VectorSpace](#) > d_space, [Number](#) &obj_scaling, SmartPtr< [Vector](#) > &x_scaling, SmartPtr< [Vector](#) > &c_scaling, SmartPtr< [Vector](#) > &d_scaling) const
Routines to get the scaling parameters.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [NLP](#) (const [NLP](#) &)
Copy Constructor.
- void [operator=](#) (const [NLP](#) &)
Overloaded Equals Operator.

6.115.1 Detailed Description

Brief Class Description.

Detailed Class Description.

Definition at line 31 of file [IpNLP.hpp](#).

6.115.2 Constructor & Destructor Documentation

6.115.2.1 [Ipopt::NLP::NLP \(\)](#) [\[inline\]](#)

Default constructor.

Definition at line 37 of file [IpNLP.hpp](#).

6.115.2.2 [virtual Ipopt::NLP::~~NLP \(\)](#) [\[inline\]](#), [\[virtual\]](#)

Default destructor.

Definition at line 41 of file [IpNLP.hpp](#).

6.115.2.3 [Ipopt::NLP::NLP \(const \[NLP\]\(#\) & \)](#) [\[private\]](#)

Copy Constructor.

6.115.3 Member Function Documentation

6.115.3.1 [Ipopt::NLP::DECLARE_STD_EXCEPTION \(USER_SCALING_NOT_IMPLEMENTED \)](#)

Exceptions.

6.115.3.2 [Ipopt::NLP::DECLARE_STD_EXCEPTION \(INVALID_NLP \)](#)

6.115.3.3 [virtual bool Ipopt::NLP::ProcessOptions \(const OptionsList & *options*, const std::string & *prefix* \)](#) [\[inline\]](#), [\[virtual\]](#)

Overload if you want the chance to process options or parameters that may be specific to the [NLP](#).

Reimplemented in [Ipopt::TNLPAdapter](#), and [Ipopt::NLPBoundsRemover](#).

Definition at line 56 of file [IpNLP.hpp](#).

6.115.3.4 `virtual bool Ipopt::NLP::GetSpaces (SmartPtr< const VectorSpace > & x_space, SmartPtr< const VectorSpace > & c_space, SmartPtr< const VectorSpace > & d_space, SmartPtr< const VectorSpace > & x_l_space, SmartPtr< const MatrixSpace > & px_l_space, SmartPtr< const VectorSpace > & x_u_space, SmartPtr< const MatrixSpace > & px_u_space, SmartPtr< const VectorSpace > & d_l_space, SmartPtr< const MatrixSpace > & pd_l_space, SmartPtr< const VectorSpace > & d_u_space, SmartPtr< const MatrixSpace > & pd_u_space, SmartPtr< const MatrixSpace > & Jac_c_space, SmartPtr< const MatrixSpace > & Jac_d_space, SmartPtr< const SymMatrixSpace > & Hess_lagrangian_space) [pure virtual]`

Method for creating the derived vector / matrix types.

The Hess_lagrangian_space pointer can be NULL if a quasi-Newton options is chosen.

Implemented in [Ipopt::TNLPAdapter](#), and [Ipopt::NLPBoundsRemover](#).

6.115.3.5 `virtual bool Ipopt::NLP::GetBoundsInformation (const Matrix & Px_L, Vector & x_L, const Matrix & Px_U, Vector & x_U, const Matrix & Pd_L, Vector & d_L, const Matrix & Pd_U, Vector & d_U) [pure virtual]`

Method for obtaining the bounds information.

Implemented in [Ipopt::TNLPAdapter](#), and [Ipopt::NLPBoundsRemover](#).

6.115.3.6 `virtual bool Ipopt::NLP::GetStartingPoint (SmartPtr< Vector > x, bool need_x, SmartPtr< Vector > y_c, bool need_y_c, SmartPtr< Vector > y_d, bool need_y_d, SmartPtr< Vector > z_L, bool need_z_L, SmartPtr< Vector > z_U, bool need_z_U) [pure virtual]`

Method for obtaining the starting point for all the iterates.

ToDo it might not make sense to ask for initial values for v_L and v_U?

Implemented in [Ipopt::TNLPAdapter](#), and [Ipopt::NLPBoundsRemover](#).

6.115.3.7 `virtual bool Ipopt::NLP::GetWarmStartIterate (IteratesVector & warm_start_iterate) [inline],[virtual]`

Method for obtaining an entire iterate as a warmstart point.

The incoming [IteratesVector](#) has to be filled. The default dummy implementation returns false.

Reimplemented in [Ipopt::TNLPAdapter](#), and [Ipopt::NLPBoundsRemover](#).

Definition at line 109 of file IpNLP.hpp.

6.115.3.8 `virtual bool Ipopt::NLP::Eval_f (const Vector & x, Number & f) [pure virtual]`

Implemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

6.115.3.9 `virtual bool Ipopt::NLP::Eval_grad_f (const Vector & x, Vector & g_f) [pure virtual]`

Implemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

6.115.3.10 `virtual bool Ipopt::NLP::Eval_c (const Vector & x, Vector & c) [pure virtual]`

Implemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

6.115.3.11 `virtual bool Ipopt::NLP::Eval_jac_c (const Vector & x, Matrix & jac_c) [pure virtual]`

Implemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

6.115.3.12 `virtual bool Ipopt::NLP::Eval_d (const Vector & x, Vector & d) [pure virtual]`

Implemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

6.115.3.13 `virtual bool Ipopt::NLP::Eval_jac_d (const Vector & x, Matrix & jac_d) [pure virtual]`

Implemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

6.115.3.14 `virtual bool Ipopt::NLP::Eval_h (const Vector & x, Number obj_factor, const Vector & yc, const Vector & yd, SymMatrix & h) [pure virtual]`

Implemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

6.115.3.15 `virtual void Ipopt::NLP::FinalizeSolution (SolverReturn status, const Vector & x, const Vector & z_L, const Vector & z_U, const Vector & c, const Vector & d, const Vector & y_c, const Vector & y_d, Number obj_value, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [inline],[virtual]`

This method is called at the very end of the optimization.

It provides the final iterate to the user, so that it can be stored as the solution. The status flag indicates the outcome of the optimization, where SolverReturn is defined in [IpAlgTypes.hpp](#).

Reimplemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

Definition at line 145 of file IpNLP.hpp.

6.115.3.16 `virtual bool Ipopt::NLP::IntermediateCallBack (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [inline],[virtual]`

This method is called once per iteration, after the iteration summary output has been printed.

It provides the current information to the user to do with it anything she wants. It also allows the user to ask for a premature termination of the optimization by returning false, in which case [Ipopt](#) will terminate with a corresponding return status. The basic information provided in the argument list has the quantities values printed in the iteration summary line. If more information is required, a user can obtain it from the IpData and IpCalculatedQuantities objects. However, note that the provided quantities are all for the problem that [Ipopt](#) sees, i.e., the quantities might be scaled, fixed variables might be sorted out, etc. The status indicates things like whether the algorithm is in the restoration phase... In the restoration phase, the dual variables are probably not changing.

Reimplemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

Definition at line 170 of file IpNLP.hpp.

6.115.3.17 `virtual void Ipopt::NLP::GetScalingParameters (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, Number & obj_scaling, SmartPtr< Vector > & x_scaling, SmartPtr< Vector > & c_scaling, SmartPtr< Vector > & d_scaling) const [inline],[virtual]`

Routines to get the scaling parameters.

These do not need to be overloaded unless the options are set for User scaling

Reimplemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

Definition at line 188 of file IpNLP.hpp.

6.115.3.18 `virtual void Ipopt::NLP::GetQuasiNewtonApproximationSpaces (SmartPtr< VectorSpace > & approx_space, SmartPtr< Matrix > & P_approx) [inline],[virtual]`

Method for obtaining the subspace in which the limited-memory Hessian approximation should be done.

This is only called if the limited-memory Hessian approximation is chosen. Since the Hessian is zero in the space of all variables that appear in the problem functions only linearly, this allows the user to provide a [VectorSpace](#) for all

nonlinear variables, and an [ExpansionMatrix](#) to lift from this [VectorSpace](#) to the [VectorSpace](#) of the primal variables x . If the returned values are NULL, it is assumed that the Hessian is to be approximated in the space of all x variables. The default instantiation of this method returns NULL, and a user only has to overwrite this method if the approximation is to be done only in a subspace.

Reimplemented in [Ipopt::NLPBoundsRemover](#), and [Ipopt::TNLPAdapter](#).

Definition at line 217 of file IpNLP.hpp.

6.115.3.19 void Ipopt::NLP::operator= (const NLP &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

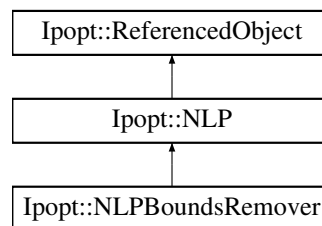
- [Interfaces/IpNLP.hpp](#)

6.116 Ipopt::NLPBoundsRemover Class Reference

This is an adapter for an [NLP](#) that converts variable bound constraints to inequality constraints.

```
#include <IpNLPBoundsRemover.hpp>
```

Inheritance diagram for Ipopt::NLPBoundsRemover:



Public Member Functions

- virtual void [GetQuasiNewtonApproximationSpaces](#) (SmartPtr< [VectorSpace](#) > &approx_space, SmartPtr< [Matrix](#) > &P_approx)

Method for obtaining the subspace in which the limited-memory Hessian approximation should be done.

- [SmartPtr< NLP > nlp](#) ()

Accessor method to the original [NLP](#).

Constructors/Destructors

- [NLPBoundsRemover](#) ([NLP](#) &nlp, bool allow_twosided_inequalities=false)

The constructor is given the [NLP](#) of which the bounds are to be replaced by inequality constraints.

- virtual [~NLPBoundsRemover](#) ()

Default destructor.

NLP Initialization (overload in

derived classes).

- virtual bool [ProcessOptions](#) (const [OptionsList](#) &options, const std::string &prefix)

Overload if you want the chance to process options or parameters that may be specific to the [NLP](#).

- virtual bool `GetSpaces` (`SmartPtr`< const `VectorSpace` > &x_space, `SmartPtr`< const `VectorSpace` > &c_space, `SmartPtr`< const `VectorSpace` > &d_space, `SmartPtr`< const `VectorSpace` > &x_l_space, `SmartPtr`< const `MatrixSpace` > &px_l_space, `SmartPtr`< const `VectorSpace` > &x_u_space, `SmartPtr`< const `MatrixSpace` > &px_u_space, `SmartPtr`< const `VectorSpace` > &d_l_space, `SmartPtr`< const `MatrixSpace` > &pd_l_space, `SmartPtr`< const `VectorSpace` > &d_u_space, `SmartPtr`< const `MatrixSpace` > &pd_u_space, `SmartPtr`< const `MatrixSpace` > &Jac_c_space, `SmartPtr`< const `MatrixSpace` > &Jac_d_space, `SmartPtr`< const `SymMatrixSpace` > &Hess_lagrangian_space)
Method for creating the derived vector / matrix types.
- virtual bool `GetBoundsInformation` (const `Matrix` &Px_L, `Vector` &x_L, const `Matrix` &Px_U, `Vector` &x_U, const `Matrix` &Pd_L, `Vector` &d_L, const `Matrix` &Pd_U, `Vector` &d_U)
Method for obtaining the bounds information.
- virtual bool `GetStartingPoint` (`SmartPtr`< `Vector` > x, bool need_x, `SmartPtr`< `Vector` > y_c, bool need_y_c, `SmartPtr`< `Vector` > y_d, bool need_y_d, `SmartPtr`< `Vector` > z_L, bool need_z_L, `SmartPtr`< `Vector` > z_U, bool need_z_U)
Method for obtaining the starting point for all the iterates.
- virtual bool `GetWarmStartIterate` (`IteratesVector` &warm_start_iterate)
Method for obtaining an entire iterate as a warmstart point.

NLP evaluation routines (overload

in derived classes.

- virtual bool `Eval_f` (const `Vector` &x, `Number` &f)
- virtual bool `Eval_grad_f` (const `Vector` &x, `Vector` &g_f)
- virtual bool `Eval_c` (const `Vector` &x, `Vector` &c)
- virtual bool `Eval_jac_c` (const `Vector` &x, `Matrix` &jac_c)
- virtual bool `Eval_d` (const `Vector` &x, `Vector` &d)
- virtual bool `Eval_jac_d` (const `Vector` &x, `Matrix` &jac_d)
- virtual bool `Eval_h` (const `Vector` &x, `Number` obj_factor, const `Vector` &yc, const `Vector` &y_d, `SymMatrix` &h)

NLP solution routines. Have default dummy

implementations that can be overloaded.

- virtual void `FinalizeSolution` (`SolverReturn` status, const `Vector` &x, const `Vector` &z_L, const `Vector` &z_U, const `Vector` &c, const `Vector` &d, const `Vector` &y_c, const `Vector` &y_d, `Number` obj_value, const `IpoptData` *ip_data, `IpoptCalculatedQuantities` *ip_cq)
This method is called at the very end of the optimization.
- virtual bool `IntermediateCallBack` (`AlgorithmMode` mode, `Index` iter, `Number` obj_value, `Number` inf_pr, `Number` inf_du, `Number` mu, `Number` d_norm, `Number` regularization_size, `Number` alpha_du, `Number` alpha_pr, `Index` ls_trials, const `IpoptData` *ip_data, `IpoptCalculatedQuantities` *ip_cq)
This method is called once per iteration, after the iteration summary output has been printed.
- virtual void `GetScalingParameters` (const `SmartPtr`< const `VectorSpace` > x_space, const `SmartPtr`< const `VectorSpace` > c_space, const `SmartPtr`< const `VectorSpace` > d_space, `Number` &obj_scaling, `SmartPtr`< `Vector` > &x_scaling, `SmartPtr`< `Vector` > &c_scaling, `SmartPtr`< `Vector` > &d_scaling) const
Routines to get the scaling parameters.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [NLPBoundsRemover](#) ()
Default Constructor.
- [NLPBoundsRemover](#) (const [NLPBoundsRemover](#) &)
Copy Constructor.
- void [operator=](#) (const [NLPBoundsRemover](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [NLP](#) > [nlp_](#)
Pointer to the original [NLP](#).
- [SmartPtr](#)< const [Matrix](#) > [Px_l_orig_](#)
Pointer to the expansion matrix for the lower x bounds.
- [SmartPtr](#)< const [Matrix](#) > [Px_u_orig_](#)
Pointer to the expansion matrix for the upper x bounds.
- [SmartPtr](#)< const [VectorSpace](#) > [d_space_orig_](#)
Pointer to the original d space.
- bool [allow_twosided_inequalities_](#)
Flag indicating whether twosided inequality constraints are allowed.

6.116.1 Detailed Description

This is an adapter for an [NLP](#) that converts variable bound constraints to inequality constraints.

This is necessary for the version of [Ipopt](#) that uses iterative linear solvers. At this point, none of the original inequality constraints is allowed to have both lower and upper bounds. The [NLP](#) visible to [Ipopt](#) via this adapter will not have any bounds on variables, but have equivalent inequality constraints.

Definition at line 24 of file [IpNLPBoundsRemover.hpp](#).

6.116.2 Constructor & Destructor Documentation

6.116.2.1 [Ipopt::NLPBoundsRemover::NLPBoundsRemover](#) ([NLP](#) & *nlp*, bool *allow_twosided_inequalities* = false)

The constructor is given the [NLP](#) of which the bounds are to be replaced by inequality constraints.

6.116.2.2 [virtual Ipopt::NLPBoundsRemover::~~NLPBoundsRemover](#) () [\[inline\]](#),[\[virtual\]](#)

Default destructor.

Definition at line 34 of file [IpNLPBoundsRemover.hpp](#).

6.116.2.3 [Ipopt::NLPBoundsRemover::NLPBoundsRemover](#) () [\[private\]](#)

Default Constructor.

6.116.2.4 [Ipopt::NLPBoundsRemover::NLPBoundsRemover](#) (const [NLPBoundsRemover](#) &) [\[private\]](#)

Copy Constructor.

6.116.3 Member Function Documentation

6.116.3.1 `virtual bool lpopt::NLPBoundsRemover::ProcessOptions (const OptionsList & options, const std::string & prefix) [inline],[virtual]`

Overload if you want the chance to process options or parameters that may be specific to the [NLP](#).

Reimplemented from [lpopt::NLP](#).

Definition at line 43 of file `lpNLPBoundsRemover.hpp`.

6.116.3.2 `virtual bool lpopt::NLPBoundsRemover::GetSpaces (SmartPtr< const VectorSpace > & x_space, SmartPtr< const VectorSpace > & c_space, SmartPtr< const VectorSpace > & d_space, SmartPtr< const VectorSpace > & x_L_space, SmartPtr< const MatrixSpace > & px_L_space, SmartPtr< const VectorSpace > & x_U_space, SmartPtr< const MatrixSpace > & px_U_space, SmartPtr< const VectorSpace > & d_L_space, SmartPtr< const MatrixSpace > & pd_L_space, SmartPtr< const VectorSpace > & d_U_space, SmartPtr< const MatrixSpace > & pd_U_space, SmartPtr< const MatrixSpace > & Jac_c_space, SmartPtr< const MatrixSpace > & Jac_d_space, SmartPtr< const SymMatrixSpace > & Hess_lagrangian_space) [virtual]`

Method for creating the derived vector / matrix types.

The `Hess_lagrangian_space` pointer can be NULL if a quasi-Newton options is chosen.

Implements [lpopt::NLP](#).

6.116.3.3 `virtual bool lpopt::NLPBoundsRemover::GetBoundsInformation (const Matrix & Px_L, Vector & x_L, const Matrix & Px_U, Vector & x_U, const Matrix & Pd_L, Vector & d_L, const Matrix & Pd_U, Vector & d_U) [virtual]`

Method for obtaining the bounds information.

Implements [lpopt::NLP](#).

6.116.3.4 `virtual bool lpopt::NLPBoundsRemover::GetStartingPoint (SmartPtr< Vector > x, bool need_x, SmartPtr< Vector > y_c, bool need_y_c, SmartPtr< Vector > y_d, bool need_y_d, SmartPtr< Vector > z_L, bool need_z_L, SmartPtr< Vector > z_U, bool need_z_U) [virtual]`

Method for obtaining the starting point for all the iterates.

ToDo it might not make sense to ask for initial values for `v_L` and `v_U`?

Implements [lpopt::NLP](#).

6.116.3.5 `virtual bool lpopt::NLPBoundsRemover::GetWarmStartIterate (IteratesVector & warm_start_iterate) [inline],[virtual]`

Method for obtaining an entire iterate as a warmstart point.

The incoming [IteratesVector](#) has to be filled. This has not yet been implemented for this adapter.

Reimplemented from [lpopt::NLP](#).

Definition at line 94 of file `lpNLPBoundsRemover.hpp`.

6.116.3.6 `virtual bool lpopt::NLPBoundsRemover::Eval_f (const Vector & x, Number & f) [inline],[virtual]`

Implements [lpopt::NLP](#).

Definition at line 103 of file `lpNLPBoundsRemover.hpp`.

6.116.3.7 `virtual bool Ipopt::NLPBoundsRemover::Eval_grad_f(const Vector & x, Vector & g_f) [inline],[virtual]`

Implements [Ipopt::NLP](#).

Definition at line 108 of file IpNLPBoundsRemover.hpp.

6.116.3.8 `virtual bool Ipopt::NLPBoundsRemover::Eval_c(const Vector & x, Vector & c) [inline],[virtual]`

Implements [Ipopt::NLP](#).

Definition at line 113 of file IpNLPBoundsRemover.hpp.

6.116.3.9 `virtual bool Ipopt::NLPBoundsRemover::Eval_jac_c(const Vector & x, Matrix & jac_c) [inline],[virtual]`

Implements [Ipopt::NLP](#).

Definition at line 118 of file IpNLPBoundsRemover.hpp.

6.116.3.10 `virtual bool Ipopt::NLPBoundsRemover::Eval_d(const Vector & x, Vector & d) [virtual]`

Implements [Ipopt::NLP](#).

6.116.3.11 `virtual bool Ipopt::NLPBoundsRemover::Eval_jac_d(const Vector & x, Matrix & jac_d) [virtual]`

Implements [Ipopt::NLP](#).

6.116.3.12 `virtual bool Ipopt::NLPBoundsRemover::Eval_h(const Vector & x, Number obj_factor, const Vector & yc, const Vector & yd, SymMatrix & h) [virtual]`

Implements [Ipopt::NLP](#).

6.116.3.13 `virtual void Ipopt::NLPBoundsRemover::FinalizeSolution(SolverReturn status, const Vector & x, const Vector & z_L, const Vector & z_U, const Vector & c, const Vector & d, const Vector & y_c, const Vector & y_d, Number obj_value, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [virtual]`

This method is called at the very end of the optimization.

It provides the final iterate to the user, so that it can be stored as the solution. The status flag indicates the outcome of the optimization, where SolverReturn is defined in [IpAlgTypes.hpp](#).

Reimplemented from [Ipopt::NLP](#).

6.116.3.14 `virtual bool Ipopt::NLPBoundsRemover::IntermediateCallBack(AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [inline],[virtual]`

This method is called once per iteration, after the iteration summary output has been printed.

It provides the current information to the user to do with it anything she wants. It also allows the user to ask for a premature termination of the optimization by returning false, in which case [Ipopt](#) will terminate with a corresponding return status. The basic information provided in the argument list has the quantities values printed in the iteration summary line. If more information is required, a user can obtain it from the IpData and IpCalculatedQuantities objects. However, note that the provided quantities are all for the problem that [Ipopt](#) sees, i.e., the quantities might be scaled, fixed variables might be sorted out, etc. The status indicates things like whether the algorithm is in the restoration phase... In the restoration phase, the dual variables are probably not changing.

Reimplemented from [Ipopt::NLP](#).

Definition at line 166 of file IpNLPBoundsRemover.hpp.

6.116.3.15 `virtual void Ipopt::NLPBoundsRemover::GetScalingParameters (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, Number & obj_scaling, SmartPtr< Vector > & x_scaling, SmartPtr< Vector > & c_scaling, SmartPtr< Vector > & d_scaling) const [virtual]`

Routines to get the scaling parameters.

These do not need to be overloaded unless the options are set for User scaling

Reimplemented from [Ipopt::NLP](#).

6.116.3.16 `virtual void Ipopt::NLPBoundsRemover::GetQuasiNewtonApproximationSpaces (SmartPtr< VectorSpace > & approx_space, SmartPtr< Matrix > & P_approx) [inline],[virtual]`

Method for obtaining the subspace in which the limited-memory Hessian approximation should be done.

This is only called if the limited-memory Hessian approximation is chosen. Since the Hessian is zero in the space of all variables that appear in the problem functions only linearly, this allows the user to provide a [VectorSpace](#) for all nonlinear variables, and an [ExpansionMatrix](#) to lift from this [VectorSpace](#) to the [VectorSpace](#) of the primal variables x. If the returned values are NULL, it is assumed that the Hessian is to be approximated in the space of all x variables. The default instantiation of this method returns NULL, and a user only has to overwrite this method if the approximation is to be done only in a subspace.

Reimplemented from [Ipopt::NLP](#).

Definition at line 211 of file `IpNLPBoundsRemover.hpp`.

6.116.3.17 `SmartPtr<NLP> Ipopt::NLPBoundsRemover::nlp () [inline]`

Accessor method to the original [NLP](#).

Definition at line 218 of file `IpNLPBoundsRemover.hpp`.

6.116.3.18 `void Ipopt::NLPBoundsRemover::operator= (const NLPBoundsRemover &) [private]`

Overloaded Equals Operator.

6.116.4 Member Data Documentation

6.116.4.1 `SmartPtr<NLP> Ipopt::NLPBoundsRemover::nlp_ [private]`

Pointer to the original [NLP](#).

Definition at line 242 of file `IpNLPBoundsRemover.hpp`.

6.116.4.2 `SmartPtr<const Matrix> Ipopt::NLPBoundsRemover::Px_l_orig_ [private]`

Pointer to the expansion matrix for the lower x bounds.

Definition at line 245 of file `IpNLPBoundsRemover.hpp`.

6.116.4.3 `SmartPtr<const Matrix> Ipopt::NLPBoundsRemover::Px_u_orig_ [private]`

Pointer to the expansion matrix for the upper x bounds.

Definition at line 248 of file `IpNLPBoundsRemover.hpp`.

6.116.4.4 `SmartPtr<const VectorSpace> Ipopt::NLPBoundsRemover::d_space_orig_ [private]`

Pointer to the original d space.

Definition at line 251 of file IpNLPBoundsRemover.hpp.

6.116.4.5 `bool Ipopt::NLPBoundsRemover::allow_twosided_inequalities_ [private]`

Flag indicating whether twosided inequality constraints are allowed.

Definition at line 255 of file IpNLPBoundsRemover.hpp.

The documentation for this class was generated from the following file:

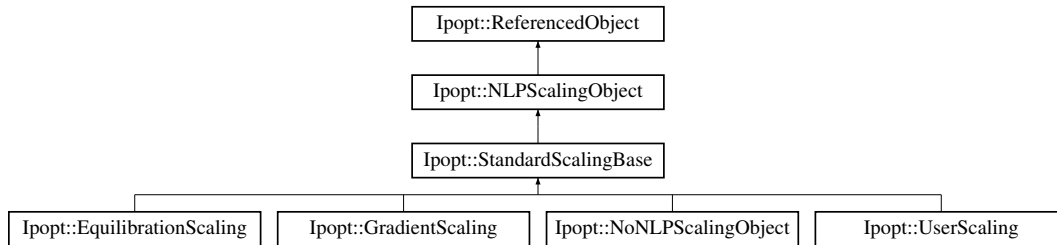
- Algorithm/IpNLPBoundsRemover.hpp

6.117 Ipopt::NLPScalingObject Class Reference

This is the abstract base class for problem scaling.

```
#include <IpNLPScaling.hpp>
```

Inheritance diagram for Ipopt::NLPScalingObject:



Public Member Functions

- `bool Initialize (const Journalist &jnlst, const OptionsList &options, const std::string &prefix)`
Method to initialize the options.
- `virtual void DetermineScaling (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, SmartPtr< const MatrixSpace > &new_jac_c_space, SmartPtr< const MatrixSpace > &new_jac_d_space, SmartPtr< const SymMatrixSpace > &new_h_space, const Matrix &Px_L, const Vector &x_L, const Matrix &Px_U, const Vector &x_U)=0`

This method is called by the `IpoptNLP`'s at a convenient time to compute and/or read scaling factors.

Constructors/Destructors

- `NLPScalingObject ()`
- `virtual ~NLPScalingObject ()`
Default destructor.
- `virtual Number apply_obj_scaling (const Number &f)=0`
Methods to map scaled and unscaled matrices.
- `virtual Number unapply_obj_scaling (const Number &f)=0`
Returns an obj-unscaled version of the given scalar.
- `virtual SmartPtr< Vector > apply_vector_scaling_x_NonConst (const SmartPtr< const Vector > &v)=0`
Returns an x-scaled version of the given vector.

- virtual `SmartPtr< const Vector > apply_vector_scaling_x` (const `SmartPtr< const Vector >` &v)=0
Returns an x-scaled version of the given vector.
- virtual `SmartPtr< Vector > unapply_vector_scaling_x_NonConst` (const `SmartPtr< const Vector >` &v)=0
Returns an x-unscaled version of the given vector.
- virtual `SmartPtr< const Vector > unapply_vector_scaling_x` (const `SmartPtr< const Vector >` &v)=0
Returns an x-unscaled version of the given vector.
- virtual `SmartPtr< const Vector > apply_vector_scaling_c` (const `SmartPtr< const Vector >` &v)=0
Returns an c-scaled version of the given vector.
- virtual `SmartPtr< const Vector > unapply_vector_scaling_c` (const `SmartPtr< const Vector >` &v)=0
Returns an c-unscaled version of the given vector.
- virtual `SmartPtr< Vector > apply_vector_scaling_c_NonConst` (const `SmartPtr< const Vector >` &v)=0
Returns an c-scaled version of the given vector.
- virtual `SmartPtr< Vector > unapply_vector_scaling_c_NonConst` (const `SmartPtr< const Vector >` &v)=0
Returns an c-unscaled version of the given vector.
- virtual `SmartPtr< const Vector > apply_vector_scaling_d` (const `SmartPtr< const Vector >` &v)=0
Returns an d-scaled version of the given vector.
- virtual `SmartPtr< const Vector > unapply_vector_scaling_d` (const `SmartPtr< const Vector >` &v)=0
Returns an d-unscaled version of the given vector.
- virtual `SmartPtr< Vector > apply_vector_scaling_d_NonConst` (const `SmartPtr< const Vector >` &v)=0
Returns an d-scaled version of the given vector.
- virtual `SmartPtr< Vector > unapply_vector_scaling_d_NonConst` (const `SmartPtr< const Vector >` &v)=0
Returns an d-unscaled version of the given vector.
- virtual `SmartPtr< const Matrix > apply_jac_c_scaling` (`SmartPtr< const Matrix >` matrix)=0
Returns a scaled version of the jacobian for c.
- virtual `SmartPtr< const Matrix > apply_jac_d_scaling` (`SmartPtr< const Matrix >` matrix)=0
Returns a scaled version of the jacobian for d If the overloaded method does not create a new matrix, make sure to set the matrix ptr passed in to NULL.
- virtual `SmartPtr< const SymMatrix > apply_hessian_scaling` (`SmartPtr< const SymMatrix >` matrix)=0
Returns a scaled version of the hessian of the lagrangian If the overloaded method does not create a new matrix, make sure to set the matrix ptr passed in to NULL.
- `SmartPtr< Vector > apply_vector_scaling_x_LU_NonConst` (const `Matrix &`Px_LU, const `SmartPtr< const Vector >` &lu, const `VectorSpace &`x_space)
Methods for scaling bounds - these wrap those above.
- `SmartPtr< const Vector > apply_vector_scaling_x_LU` (const `Matrix &`Px_LU, const `SmartPtr< const Vector >` &lu, const `VectorSpace &`x_space)
Returns an x-scaled vector in the x_L or x_U space.
- `SmartPtr< Vector > apply_vector_scaling_d_LU_NonConst` (const `Matrix &`Pd_LU, const `SmartPtr< const Vector >` &lu, const `VectorSpace &`d_space)
Returns an d-scaled vector in the d_L or d_U space.
- `SmartPtr< const Vector > apply_vector_scaling_d_LU` (const `Matrix &`Pd_LU, const `SmartPtr< const Vector >` &lu, const `VectorSpace &`d_space)
Returns an d-scaled vector in the d_L or d_U space.
- `SmartPtr< Vector > unapply_vector_scaling_d_LU_NonConst` (const `Matrix &`Pd_LU, const `SmartPtr< const Vector >` &lu, const `VectorSpace &`d_space)
Returns an d-unscaled vector in the d_L or d_U space.
- `SmartPtr< const Vector > unapply_vector_scaling_d_LU` (const `Matrix &`Pd_LU, const `SmartPtr< const Vector >` &lu, const `VectorSpace &`d_space)

Returns an d -unscaled vector in the d_L or d_U space.

- virtual `SmartPtr< Vector > apply_grad_obj_scaling_NonConst` (const `SmartPtr< const Vector >` &v)
Methods for scaling the gradient of the objective - wraps the virtual methods above.
- virtual `SmartPtr< const Vector > apply_grad_obj_scaling` (const `SmartPtr< const Vector >` &v)
Returns a $grad_f$ scaled version ($d_f * D_x^{-1}$) of the given vector.
- virtual `SmartPtr< Vector > unapply_grad_obj_scaling_NonConst` (const `SmartPtr< const Vector >` &v)
Returns a $grad_f$ unscaled version ($d_f * D_x^{-1}$) of the given vector.
- virtual `SmartPtr< const Vector > unapply_grad_obj_scaling` (const `SmartPtr< const Vector >` &v)
Returns a $grad_f$ unscaled version ($d_f * D_x^{-1}$) of the given vector.

Methods for determining whether scaling for entities is

done

- virtual bool `have_x_scaling` ()=0
Returns true if the primal x variables are scaled.
- virtual bool `have_c_scaling` ()=0
Returns true if the equality constraints are scaled.
- virtual bool `have_d_scaling` ()=0
Returns true if the inequality constraints are scaled.

Protected Member Functions

- virtual bool `InitializeImpl` (const `OptionsList` &options, const `std::string` &prefix)=0
Implementation of the initialization method that has to be overloaded by for each derived class.
- const `Journalist` & `Jnlst` () const
Accessor method for the journalist.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `NLPScalingObject` (const `NLPScalingObject` &)
Copy Constructor.
- void `operator=` (const `NLPScalingObject` &)
Overloaded Equals Operator.

Private Attributes

- `SmartPtr< const Journalist >` `jnlst_`

6.117.1 Detailed Description

This is the abstract base class for problem scaling.

It is responsible for determining the scaling factors and mapping quantities in and out of scaled and unscaled versions

Definition at line 32 of file `IpNLPScaling.hpp`.

6.117.2 Constructor & Destructor Documentation

6.117.2.1 `Ipopt::NLPScalingObject::NLPScalingObject ()`

6.117.2.2 `virtual Ipopt::NLPScalingObject::~~NLPScalingObject () [virtual]`

Default destructor.

6.117.2.3 `Ipopt::NLPScalingObject::NLPScalingObject (const NLPScalingObject &) [private]`

Copy Constructor.

6.117.3 Member Function Documentation

6.117.3.1 `bool Ipopt::NLPScalingObject::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [inline]`

Method to initialize the options.

Definition at line 44 of file `IpNLPScaling.hpp`.

6.117.3.2 `virtual Number Ipopt::NLPScalingObject::apply_obj_scaling (const Number & f) [pure virtual]`

Methods to map scaled and unscaled matrices.

Returns an obj-scaled version of the given scalar

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.3 `virtual Number Ipopt::NLPScalingObject::unapply_obj_scaling (const Number & f) [pure virtual]`

Returns an obj-unscaled version of the given scalar.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.4 `virtual SmartPtr<Vector> Ipopt::NLPScalingObject::apply_vector_scaling_x_NonConst (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an x-scaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.5 `virtual SmartPtr<const Vector> Ipopt::NLPScalingObject::apply_vector_scaling_x (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an x-scaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.6 `virtual SmartPtr<Vector> Ipopt::NLPScalingObject::unapply_vector_scaling_x_NonConst (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an x-unscaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.7 `virtual SmartPtr<const Vector> Ipopt::NLPScalingObject::unapply_vector_scaling_x (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an x-unscaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.8 `virtual SmartPtr<const Vector> Ipopt::NLPScalingObject::apply_vector_scaling_c (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an c-scaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.9 `virtual SmartPtr<const Vector> Ipopt::NLPScalingObject::unapply_vector_scaling_c (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an c-unscaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.10 `virtual SmartPtr<Vector> Ipopt::NLPScalingObject::apply_vector_scaling_c_NonConst (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an c-scaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.11 `virtual SmartPtr<Vector> Ipopt::NLPScalingObject::unapply_vector_scaling_c_NonConst (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an c-unscaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.12 `virtual SmartPtr<const Vector> Ipopt::NLPScalingObject::apply_vector_scaling_d (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an d-scaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.13 `virtual SmartPtr<const Vector> Ipopt::NLPScalingObject::unapply_vector_scaling_d (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an d-unscaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.14 `virtual SmartPtr<Vector> Ipopt::NLPScalingObject::apply_vector_scaling_d_NonConst (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an d-scaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.15 `virtual SmartPtr<Vector> Ipopt::NLPScalingObject::unapply_vector_scaling_d_NonConst (const SmartPtr< const Vector> & v) [pure virtual]`

Returns an d-unscaled version of the given vector.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.16 **virtual SmartPtr<const Matrix> Ipopt::NLPScalingObject::apply_jac_c_scaling (SmartPtr< const Matrix > matrix)** [pure virtual]

Returns a scaled version of the jacobian for c.

If the overloaded method does not make a new matrix, make sure to set the matrix ptr passed in to NULL.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.17 **virtual SmartPtr<const Matrix> Ipopt::NLPScalingObject::apply_jac_d_scaling (SmartPtr< const Matrix > matrix)** [pure virtual]

Returns a scaled version of the jacobian for d If the overloaded method does not create a new matrix, make sure to set the matrix ptr passed in to NULL.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.18 **virtual SmartPtr<const SymMatrix> Ipopt::NLPScalingObject::apply_hessian_scaling (SmartPtr< const SymMatrix > matrix)** [pure virtual]

Returns a scaled version of the hessian of the lagrangian If the overloaded method does not create a new matrix, make sure to set the matrix ptr passed in to NULL.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.19 **SmartPtr<Vector> Ipopt::NLPScalingObject::apply_vector_scaling_x_LU_NonConst (const Matrix & Px_LU, const SmartPtr< const Vector > & lu, const VectorSpace & x_space)**

Methods for scaling bounds - these wrap those above.

Returns an x-scaled vector in the x_L or x_U space

6.117.3.20 **SmartPtr<const Vector> Ipopt::NLPScalingObject::apply_vector_scaling_x_LU (const Matrix & Px_LU, const SmartPtr< const Vector > & lu, const VectorSpace & x_space)**

Returns an x-scaled vector in the x_L or x_U space.

6.117.3.21 **SmartPtr<Vector> Ipopt::NLPScalingObject::apply_vector_scaling_d_LU_NonConst (const Matrix & Pd_LU, const SmartPtr< const Vector > & lu, const VectorSpace & d_space)**

Returns an d-scaled vector in the d_L or d_U space.

6.117.3.22 **SmartPtr<const Vector> Ipopt::NLPScalingObject::apply_vector_scaling_d_LU (const Matrix & Pd_LU, const SmartPtr< const Vector > & lu, const VectorSpace & d_space)**

Returns an d-scaled vector in the d_L or d_U space.

6.117.3.23 **SmartPtr<Vector> Ipopt::NLPScalingObject::unapply_vector_scaling_d_LU_NonConst (const Matrix & Pd_LU, const SmartPtr< const Vector > & lu, const VectorSpace & d_space)**

Returns an d-unscaled vector in the d_L or d_U space.

6.117.3.24 **SmartPtr<const Vector> Ipopt::NLPScalingObject::unapply_vector_scaling_d_LU (const Matrix & Pd_LU, const SmartPtr< const Vector > & lu, const VectorSpace & d_space)**

Returns an d-unscaled vector in the d_L or d_U space.

6.117.3.25 `virtual SmartPtr<Vector> Ipopt::NLPScalingObject::apply_grad_obj_scaling_NonConst (const SmartPtr< const Vector > & v) [virtual]`

Methods for scaling the gradient of the objective - wraps the virtual methods above.

Returns a `grad_f` scaled version ($d_f * D_x^{-1}$) of the given vector

6.117.3.26 `virtual SmartPtr<const Vector> Ipopt::NLPScalingObject::apply_grad_obj_scaling (const SmartPtr< const Vector > & v) [virtual]`

Returns a `grad_f` scaled version ($d_f * D_x^{-1}$) of the given vector.

6.117.3.27 `virtual SmartPtr<Vector> Ipopt::NLPScalingObject::unapply_grad_obj_scaling_NonConst (const SmartPtr< const Vector > & v) [virtual]`

Returns a `grad_f` unscaled version ($d_f * D_x^{-1}$) of the given vector.

6.117.3.28 `virtual SmartPtr<const Vector> Ipopt::NLPScalingObject::unapply_grad_obj_scaling (const SmartPtr< const Vector > & v) [virtual]`

Returns a `grad_f` unscaled version ($d_f * D_x^{-1}$) of the given vector.

6.117.3.29 `virtual bool Ipopt::NLPScalingObject::have_x_scaling () [pure virtual]`

Returns true if the primal x variables are scaled.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.30 `virtual bool Ipopt::NLPScalingObject::have_c_scaling () [pure virtual]`

Returns true if the equality constraints are scaled.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.31 `virtual bool Ipopt::NLPScalingObject::have_d_scaling () [pure virtual]`

Returns true if the inequality constraints are scaled.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.32 `virtual void Ipopt::NLPScalingObject::DetermineScaling (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, SmartPtr< const MatrixSpace > & new_jac_c_space, SmartPtr< const MatrixSpace > & new_jac_d_space, SmartPtr< const SymMatrixSpace > & new_h_space, const Matrix & Px_L, const Vector & x_L, const Matrix & Px_U, const Vector & x_U) [pure virtual]`

This method is called by the [IpoptNLP](#)'s at a convenient time to compute and/or read scaling factors.

Implemented in [Ipopt::StandardScalingBase](#).

6.117.3.33 `virtual bool Ipopt::NLPScalingObject::InitializImpl (const OptionsList & options, const std::string & prefix) [protected], [pure virtual]`

Implementation of the initialization method that has to be overloaded by for each derived class.

Implemented in [Ipopt::StandardScalingBase](#), [Ipopt::EquilibrationScaling](#), and [Ipopt::GradientScaling](#).

6.117.3.34 `const Journalist& Ipopt::NLPScalingObject::Jnlst () const` `[inline],[protected]`

Accessor method for the journalist.

Definition at line 200 of file `IpNLPScaling.hpp`.

6.117.3.35 `void Ipopt::NLPScalingObject::operator= (const NLPScalingObject &)` `[private]`

Overloaded Equals Operator.

6.117.4 Member Data Documentation

6.117.4.1 `SmartPtr<const Journalist> Ipopt::NLPScalingObject::jnlst_` `[private]`

Definition at line 222 of file `IpNLPScaling.hpp`.

The documentation for this class was generated from the following file:

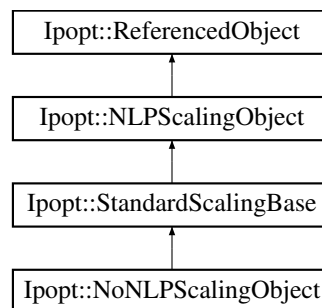
- [Algorithm/IpNLPScaling.hpp](#)

6.118 Ipopt::NoNLPScalingObject Class Reference

Class implementing the scaling object that doesn't to any scaling.

```
#include <IpNLPScaling.hpp>
```

Inheritance diagram for `Ipopt::NoNLPScalingObject`:



Public Member Functions

Constructors/Destructors

- `NoNLPScalingObject ()`
- `virtual ~NoNLPScalingObject ()`
Default destructor.

Protected Member Functions

- `virtual void DetermineScalingParametersImpl (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, const Matrix &Px_L, const Vector &x_L, const Matrix &Px_U, const Vector &x_U, Number &df, SmartPtr< Vector > &dx, SmartPtr< Vector > &dc, SmartPtr< Vector > &dd)`

Overloaded from [StandardScalingBase](#).

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [NoNLPScalingObject](#) (const [NoNLPScalingObject](#) &)
Copy Constructor.
- void `operator=` (const [NoNLPScalingObject](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.118.1 Detailed Description

Class implementing the scaling object that doesn't to any scaling.

Definition at line 400 of file `IpNLPScaling.hpp`.

6.118.2 Constructor & Destructor Documentation

6.118.2.1 `Ipopt::NoNLPScalingObject::NoNLPScalingObject () [inline]`

Definition at line 405 of file `IpNLPScaling.hpp`.

6.118.2.2 `virtual Ipopt::NoNLPScalingObject::~~NoNLPScalingObject () [inline], [virtual]`

Default destructor.

Definition at line 409 of file `IpNLPScaling.hpp`.

6.118.2.3 `Ipopt::NoNLPScalingObject::NoNLPScalingObject (const NoNLPScalingObject &) [private]`

Copy Constructor.

6.118.3 Member Function Documentation

6.118.3.1 `virtual void Ipopt::NoNLPScalingObject::DetermineScalingParametersImpl (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, const Matrix & Px_L, const Vector & x_L, const Matrix & Px_U, const Vector & x_U, Number & df, SmartPtr< Vector > & dx, SmartPtr< Vector > & dc, SmartPtr< Vector > & dd) [protected], [virtual]`

Overloaded from [StandardScalingBase](#).

Implements [Ipopt::StandardScalingBase](#).

6.118.3.2 void Ipopt::NonLPScalingObject::operator= (const NonLPScalingObject &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

- Algorithm/[IpNLPScaling.hpp](#)

6.119 Ipopt::Observer Class Reference

Slight Variation of the [Observer](#) Design Pattern.

```
#include <IpObserver.hpp>
```

Inheritance diagram for Ipopt::Observer:



Public Types

- enum [NotifyType](#) { [NT_All](#), [NT_BeingDestroyed](#), [NT_Changed](#) }
Enumeration specifying the type of notification.

Public Member Functions

Constructors/Destructors

- [Observer](#) ()
Default Constructor.
- virtual [~Observer](#) ()
Default destructor.

Protected Member Functions

- void [RequestAttach](#) ([NotifyType](#) notify_type, const [Subject](#) *subject)
Derived classes should call this method to request an "Attach" to a [Subject](#).
- void [RequestDetach](#) ([NotifyType](#) notify_type, const [Subject](#) *subject)
Derived classes should call this method to request a "Detach" to a [Subject](#).
- virtual void [RecieveNotification](#) ([NotifyType](#) notify_type, const [Subject](#) *subject)=0
Derived classes should overload this method to recieve the requested notification from attached Subjects.

Private Member Functions

- void [ProcessNotification](#) ([NotifyType](#) notify_type, const [Subject](#) *subject)
Private Method for Recieving Notification should only be called by the friend class [Subject](#).

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [Observer](#) (const [Observer](#) &)
Copy Constructor.
- void [operator=](#) (const [Observer](#) &)
Overloaded Equals Operator.

Private Attributes

- std::vector< const [Subject](#) * > [subjects_](#)
A list of the subjects currently being observed.

Friends

- class [Subject](#)

6.119.1 Detailed Description

Slight Variation of the [Observer](#) Design Pattern.

This class implements the [Observer](#) class of the [Observer](#) Design Pattern. An [Observer](#) "Attach"es to a [Subject](#), indicating that it would like to be notified of changes in the [Subject](#). Any derived class wishing to receive notifications from a [Subject](#) should inherit off of [Observer](#) and overload the protected method, `RecieveNotification_()`.

Definition at line 39 of file `IpObserver.hpp`.

6.119.2 Member Enumeration Documentation

6.119.2.1 enum `Ipopt::Observer::NotifyType`

Enumeration specifying the type of notification.

Enumerator

NT_All
NT_BeingDestroyed
NT_Changed

Definition at line 59 of file `IpObserver.hpp`.

6.119.3 Constructor & Destructor Documentation

6.119.3.1 `Ipopt::Observer::Observer ()` `[inline]`

Default Constructor.

Definition at line 50 of file `IpObserver.hpp`.

6.119.3.2 `Ipopt::Observer::~~Observer ()` `[inline]`, `[virtual]`

Default destructor.

Definition at line 198 of file `IpObserver.hpp`.

6.119.3.3 `Ipopt::Observer::Observer (const Observer &) [private]`

Copy Constructor.

6.119.4 Member Function Documentation

6.119.4.1 `void Ipopt::Observer::RequestAttach (NotifyType notify_type, const Subject * subject) [inline], [protected]`

Derived classes should call this method to request an "Attach" to a [Subject](#).

Do not call "Attach" explicitly on the [Subject](#) since further processing is done here

Definition at line 219 of file IpObserver.hpp.

6.119.4.2 `void Ipopt::Observer::RequestDetach (NotifyType notify_type, const Subject * subject) [inline], [protected]`

Derived classes should call this method to request a "Detach" to a [Subject](#).

Do not call "Detach" explicitly on the [Subject](#) since further processing is done here

Definition at line 238 of file IpObserver.hpp.

6.119.4.3 `virtual void Ipopt::Observer::RecieveNotification (NotifyType notify_type, const Subject * subject) [protected], [pure virtual]`

Derived classes should overload this method to recieve the requested notification from attached Subjects.

Implemented in [Ipopt::DependentResult< T >](#), [Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::Vector > >](#), [Ipopt::DependentResult< Ipopt::SmartPtr< Ipopt::Vector > >](#), [Ipopt::DependentResult< Number >](#), [Ipopt::DependentResult< void * >](#), [Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::SymMatrix > >](#), and [Ipopt::DependentResult< Ipopt::SmartPtr< const Ipopt::Matrix > >](#).

6.119.4.4 `void Ipopt::Observer::operator= (const Observer &) [private]`

Overloaded Equals Operator.

6.119.4.5 `void Ipopt::Observer::ProcessNotification (NotifyType notify_type, const Subject * subject) [inline], [private]`

Private Method for Recieving Notification should only be called by the friend class [Subject](#).

This method will, in turn, call the overloaded RecieveNotification method for the derived class to process.

Definition at line 268 of file IpObserver.hpp.

6.119.5 Friends And Related Function Documentation

6.119.5.1 `friend class Subject [friend]`

Definition at line 118 of file IpObserver.hpp.

6.119.6 Member Data Documentation

6.119.6.1 `std::vector<const Subject*> Ipopt::Observer::subjects_ [private]`

A list of the subjects currently being observed.

Definition at line 107 of file IpObserver.hpp.

The documentation for this class was generated from the following file:

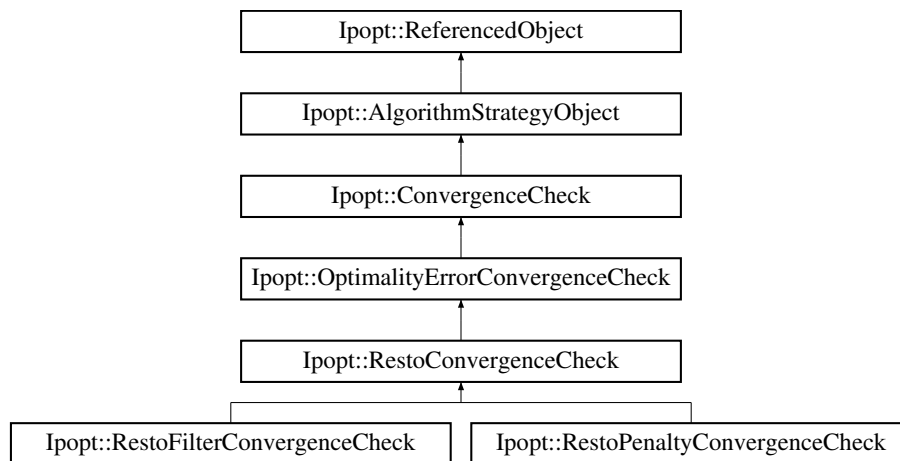
- Common/IpObserver.hpp

6.120 Ipopt::OptimalityErrorConvergenceCheck Class Reference

Brief Class Description.

```
#include <IpOptErrorConvCheck.hpp>
```

Inheritance diagram for Ipopt::OptimalityErrorConvergenceCheck:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual [ConvergenceStatus](#) [CheckConvergence](#) (bool call_intermediate_callback=true)
Overloaded convergence check.
- virtual bool [CurrentIsAcceptable](#) ()
Auxilliary function for testing whether current iterate satisfies the acceptable level of optimality.

Constructors/Destructors

- [OptimalityErrorConvergenceCheck](#) ()
Default Constructor.
- virtual [~OptimalityErrorConvergenceCheck](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (SmartPtr< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Protected Attributes

Algorithmic parameters

- [Index max_iterations_](#)
Maximal number of iterations.
- [Number dual_inf_tol_](#)
Tolerance on unscaled dual infeasibility.
- [Number constr_viol_tol_](#)
Tolerance on unscaled constraint violation.
- [Number compl_inf_tol_](#)
Tolerance on unscaled complementarity.
- [Index acceptable_iter_](#)
Number of iterations with acceptable level of accuracy, after which the algorithm terminates.
- [Number acceptable_tol_](#)
Acceptable tolerance for the problem to terminate earlier if algorithm seems stuck or cycling.
- [Number acceptable_dual_inf_tol_](#)
Acceptable tolerance on unscaled dual infeasibility.
- [Number acceptable_constr_viol_tol_](#)
Acceptable tolerance on unscaled constraint violation.
- [Number acceptable_compl_inf_tol_](#)
Acceptable tolerance on unscaled complementarity.
- [Number acceptable_obj_change_tol_](#)
Acceptable tolerance for relative objective function change from iteration to iteration.
- [Number diverging_iterates_tol_](#)
Threshold for primal iterates for divergence test.
- [Number mu_target_](#)
Desired value of the barrier parameter.
- [Number max_cpu_time_](#)
Upper bound on CPU time.

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [OptimalityErrorConvergenceCheck](#) (const [OptimalityErrorConvergenceCheck](#) &)
Copy Constructor.
- void [operator=](#) (const [OptimalityErrorConvergenceCheck](#) &)
Overloaded Equals Operator.

Private Attributes

- [Index acceptable_counter_](#)
Counter for successive iterations in which acceptability criteria are met.
- [Number last_obj_val_](#)
Value of the objective function from last iteration.
- [Number curr_obj_val_](#)
Value of the objective function from current iteration.
- [Index last_obj_val_iter_](#)
Iteration counter for which last_obj_val most recently updated.

Additional Inherited Members

6.120.1 Detailed Description

Brief Class Description.

Detailed Class Description.

Definition at line 20 of file IpOptErrorConvCheck.hpp.

6.120.2 Constructor & Destructor Documentation

6.120.2.1 Ipopt::OptimalityErrorConvergenceCheck::OptimalityErrorConvergenceCheck ()

Default Constructor.

6.120.2.2 virtual Ipopt::OptimalityErrorConvergenceCheck::~OptimalityErrorConvergenceCheck () [virtual]

Default destructor.

6.120.2.3 Ipopt::OptimalityErrorConvergenceCheck::OptimalityErrorConvergenceCheck (const OptimalityErrorConvergenceCheck &) [private]

Copy Constructor.

6.120.3 Member Function Documentation

6.120.3.1 virtual bool Ipopt::OptimalityErrorConvergenceCheck::InitializImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::ConvergenceCheck](#).

Reimplemented in [Ipopt::RestoFilterConvergenceCheck](#), [Ipopt::RestoPenaltyConvergenceCheck](#), and [Ipopt::RestoConvergenceCheck](#).

6.120.3.2 virtual ConvergenceStatus Ipopt::OptimalityErrorConvergenceCheck::CheckConvergence (bool call_intermediate_callback = true) [virtual]

Overloaded convergence check.

Implements [Ipopt::ConvergenceCheck](#).

Reimplemented in [Ipopt::RestoConvergenceCheck](#).

6.120.3.3 virtual bool Ipopt::OptimalityErrorConvergenceCheck::CurrentIsAcceptable () [virtual]

Auxilliary function for testing whether current iterate satisfies the acceptable level of optimality.

Implements [Ipopt::ConvergenceCheck](#).

6.120.3.4 static void Ipopt::OptimalityErrorConvergenceCheck::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]

Methods for IpoptType.

6.120.3.5 `void Ipopt::OptimalityErrorConvergenceCheck::operator= (const OptimalityErrorConvergenceCheck &)`
[private]

Overloaded Equals Operator.

6.120.4 Member Data Documentation

6.120.4.1 `Index Ipopt::OptimalityErrorConvergenceCheck::max_iterations_` [protected]

Maximal number of iterations.

Definition at line 53 of file IpOptErrorConvCheck.hpp.

6.120.4.2 `Number Ipopt::OptimalityErrorConvergenceCheck::dual_inf_tol_` [protected]

Tolerance on unscaled dual infeasibility.

Definition at line 55 of file IpOptErrorConvCheck.hpp.

6.120.4.3 `Number Ipopt::OptimalityErrorConvergenceCheck::constr_viol_tol_` [protected]

Tolerance on unscaled constraint violation.

Definition at line 57 of file IpOptErrorConvCheck.hpp.

6.120.4.4 `Number Ipopt::OptimalityErrorConvergenceCheck::compl_inf_tol_` [protected]

Tolerance on unscaled complementarity.

Definition at line 59 of file IpOptErrorConvCheck.hpp.

6.120.4.5 `Index Ipopt::OptimalityErrorConvergenceCheck::acceptable_iter_` [protected]

Number of iterations with acceptable level of accuracy, after which the algorithm terminates.

If 0, this heuristic is disabled.

Definition at line 63 of file IpOptErrorConvCheck.hpp.

6.120.4.6 `Number Ipopt::OptimalityErrorConvergenceCheck::acceptable_tol_` [protected]

Acceptable tolerance for the problem to terminate earlier if algorithm seems stuck or cycling.

Definition at line 66 of file IpOptErrorConvCheck.hpp.

6.120.4.7 `Number Ipopt::OptimalityErrorConvergenceCheck::acceptable_dual_inf_tol_` [protected]

Acceptable tolerance on unscaled dual infeasibility.

Definition at line 68 of file IpOptErrorConvCheck.hpp.

6.120.4.8 `Number Ipopt::OptimalityErrorConvergenceCheck::acceptable_constr_viol_tol_` [protected]

Acceptable tolerance on unscaled constraint violation.

Definition at line 70 of file IpOptErrorConvCheck.hpp.

6.120.4.9 `Number Ipopt::OptimalityErrorConvergenceCheck::acceptable_compl_inf_tol_` [protected]

Acceptable tolerance on unscaled complementarity.

Definition at line 72 of file IpOptErrorConvCheck.hpp.

6.120.4.10 Number Ipopt::OptimalityErrorConvergenceCheck::acceptable_obj_change_tol_ [protected]

Acceptable tolerance for relative objective function change from iteratoin to iteration.

Definition at line 75 of file IpOptErrorConvCheck.hpp.

6.120.4.11 Number Ipopt::OptimalityErrorConvergenceCheck::diverging_iterates_tol_ [protected]

Threshold for primal iterates for divergence test.

Definition at line 77 of file IpOptErrorConvCheck.hpp.

6.120.4.12 Number Ipopt::OptimalityErrorConvergenceCheck::mu_target_ [protected]

Desired value of the barrier parameter.

Definition at line 79 of file IpOptErrorConvCheck.hpp.

6.120.4.13 Number Ipopt::OptimalityErrorConvergenceCheck::max_cpu_time_ [protected]

Upper bound on CPU time.

Definition at line 81 of file IpOptErrorConvCheck.hpp.

6.120.4.14 Index Ipopt::OptimalityErrorConvergenceCheck::acceptable_counter_ [private]

Counter for successive iterations in which acceptability criteria are met.

Definition at line 100 of file IpOptErrorConvCheck.hpp.

6.120.4.15 Number Ipopt::OptimalityErrorConvergenceCheck::last_obj_val_ [private]

Value of the objective function from last iteration.

This is for accpetable_obj_change_tol.

Definition at line 104 of file IpOptErrorConvCheck.hpp.

6.120.4.16 Number Ipopt::OptimalityErrorConvergenceCheck::curr_obj_val_ [private]

Value of the objective function from current iteration.

This is for accpetable_obj_change_tol.

Definition at line 108 of file IpOptErrorConvCheck.hpp.

6.120.4.17 Index Ipopt::OptimalityErrorConvergenceCheck::last_obj_val_iter_ [private]

Iteration counter for which last_obj_val most recently updated.

Definition at line 111 of file IpOptErrorConvCheck.hpp.

The documentation for this class was generated from the following file:

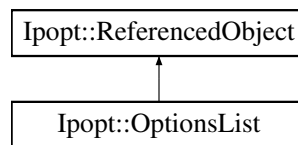
- Algorithm/[IpOptErrorConvCheck.hpp](#)

6.121 Ipopt::OptionsList Class Reference

This class stores a list of user set options.

```
#include <IpOptionsList.hpp>
```

Inheritance diagram for Ipopt::OptionsList:



Classes

- class [OptionValue](#)
Class for storing the value and counter for each option in [OptionsList](#).

Public Member Functions

- virtual void [clear](#) ()
Method for clearing all previously set options.
- virtual void [PrintList](#) (std::string &list) const
Get a string with the list of all options (tag, value, counter)
- virtual void [PrintUserOptions](#) (std::string &list) const
Get a string with the list of all options set by the user (tag, value, use/notused).
- virtual bool [ReadFromStream](#) (const [Journalist](#) &jnlst, std::istream &is)
Read options from the stream is.

Constructors/Destructors

- [OptionsList](#) (SmartPtr< [RegisteredOptions](#) > reg_options, SmartPtr< [Journalist](#) > jnlst)
- [OptionsList](#) ()
- [OptionsList](#) (const [OptionsList](#) ©)
Copy Constructor.
- virtual [~OptionsList](#) ()
Default destructor.
- virtual void [operator=](#) (const [OptionsList](#) &source)
Overloaded Equals Operator.

Get / Set Methods

- virtual void [SetRegisteredOptions](#) (const SmartPtr< [RegisteredOptions](#) > reg_options)
- virtual void [SetJournalist](#) (const SmartPtr< [Journalist](#) > jnlst)

Methods for setting options

- virtual bool [SetStringValue](#) (const std::string &tag, const std::string &value, bool allow_clobber=true, bool dont_print=false)
- virtual bool [SetNumericValue](#) (const std::string &tag, [Number](#) value, bool allow_clobber=true, bool dont_print=false)
- virtual bool [SetIntegerValue](#) (const std::string &tag, [Index](#) value, bool allow_clobber=true, bool dont_print=false)

Methods for setting options only if they have not been*set before*

- virtual bool [SetStringValueIfUnset](#) (const std::string &tag, const std::string &value, bool allow_clobber=true, bool dont_print=false)
- virtual bool [SetNumericValueIfUnset](#) (const std::string &tag, [Number](#) value, bool allow_clobber=true, bool dont_print=false)
- virtual bool [SetIntegerValueIfUnset](#) (const std::string &tag, [Index](#) value, bool allow_clobber=true, bool dont_print=false)

Methods for retrieving values from the options list. If*a tag is not found, the methods return false, and value is set to the default value defined in the registered options.*

- virtual bool [GetStringValue](#) (const std::string &tag, std::string &value, const std::string &prefix) const
- virtual bool [GetEnumValue](#) (const std::string &tag, [Index](#) &value, const std::string &prefix) const
- virtual bool [GetBoolValue](#) (const std::string &tag, bool &value, const std::string &prefix) const
- virtual bool [GetNumericValue](#) (const std::string &tag, [Number](#) &value, const std::string &prefix) const
- virtual bool [GetIntegerValue](#) (const std::string &tag, [Index](#) &value, const std::string &prefix) const

Private Member Functions

- const std::string & [lowercase](#) (const std::string tag) const
auxilliary method for converting sting to all lower-case letters
- bool [find_tag](#) (const std::string &tag, const std::string &prefix, std::string &value) const
auxilliary method for finding the value for a tag in the options list.
- bool [will_allow_clobber](#) (const std::string &tag) const
tells whether or not we can clobber a particular option.
- bool [readnexttoken](#) (std::istream &is, std::string &token)
read the next token from stream is.

Private Attributes

- std::map< std::string, [OptionValue](#) > [options_](#)
Default Constructor.
- [SmartPtr](#)< [RegisteredOptions](#) > [reg_options_](#)
list of all the registered options to validate against
- [SmartPtr](#)< [Journalist](#) > [jnlst_](#)
[Journalist](#) for writing error messages, etc.
- std::string [lowercase_buffer_](#)
auxilliary string set by lowercase method

6.121.1 Detailed Description

This class stores a list of user set options.

Each options is identified by a case-insensitive keyword (tag). Its value is stored internally as a string (always lower case), but for convenience set and get methods are provided to obtain Index and Number type values. For each keyword we also keep track of how often the value of an option has been requested by a get method.

Definition at line 32 of file IpOptionsList.hpp.

6.121.2 Constructor & Destructor Documentation

6.121.2.1 `Ipopt::OptionsList::OptionsList (SmartPtr< RegisteredOptions > reg_options, SmartPtr< Journalist > jnlst)` `[inline]`

Definition at line 142 of file `IpOptionsList.hpp`.

6.121.2.2 `Ipopt::OptionsList::OptionsList ()` `[inline]`

Definition at line 146 of file `IpOptionsList.hpp`.

6.121.2.3 `Ipopt::OptionsList::OptionsList (const OptionsList & copy)` `[inline]`

Copy Constructor.

Definition at line 150 of file `IpOptionsList.hpp`.

6.121.2.4 `virtual Ipopt::OptionsList::~OptionsList ()` `[inline],[virtual]`

Default destructor.

Definition at line 159 of file `IpOptionsList.hpp`.

6.121.3 Member Function Documentation

6.121.3.1 `virtual void Ipopt::OptionsList::operator= (const OptionsList & source)` `[inline],[virtual]`

Overloaded Equals Operator.

Definition at line 163 of file `IpOptionsList.hpp`.

6.121.3.2 `virtual void Ipopt::OptionsList::clear ()` `[inline],[virtual]`

Method for clearing all previously set options.

Definition at line 172 of file `IpOptionsList.hpp`.

6.121.3.3 `virtual void Ipopt::OptionsList::SetRegisteredOptions (const SmartPtr< RegisteredOptions > reg_options)` `[inline],[virtual]`

Definition at line 179 of file `IpOptionsList.hpp`.

6.121.3.4 `virtual void Ipopt::OptionsList::SetJournalist (const SmartPtr< Journalist > jnlst)` `[inline],[virtual]`

Definition at line 183 of file `IpOptionsList.hpp`.

6.121.3.5 `virtual bool Ipopt::OptionsList::SetStringValue (const std::string & tag, const std::string & value, bool allow_clobber = true, bool dont_print = false)` `[virtual]`

6.121.3.6 `virtual bool Ipopt::OptionsList::SetNumericValue (const std::string & tag, Number value, bool allow_clobber = true, bool dont_print = false)` `[virtual]`

6.121.3.7 `virtual bool Ipopt::OptionsList::SetIntegerValue (const std::string & tag, Index value, bool allow_clobber = true, bool dont_print = false)` `[virtual]`

6.121.3.8 `virtual bool Ipopt::OptionsList::SetStringValueIfUnset (const std::string & tag, const std::string & value, bool allow_clobber = true, bool dont_print = false)` `[virtual]`

6.121.3.9 `virtual bool Ipopt::OptionsList::SetNumericValueIfUnset (const std::string & tag, Number value, bool allow_clobber = true, bool dont_print = false) [virtual]`

6.121.3.10 `virtual bool Ipopt::OptionsList::SetIntegerValueIfUnset (const std::string & tag, Index value, bool allow_clobber = true, bool dont_print = false) [virtual]`

6.121.3.11 `virtual bool Ipopt::OptionsList::GetStringValue (const std::string & tag, std::string & value, const std::string & prefix) const [virtual]`

6.121.3.12 `virtual bool Ipopt::OptionsList::GetEnumValue (const std::string & tag, Index & value, const std::string & prefix) const [virtual]`

6.121.3.13 `virtual bool Ipopt::OptionsList::GetBoolValue (const std::string & tag, bool & value, const std::string & prefix) const [virtual]`

6.121.3.14 `virtual bool Ipopt::OptionsList::GetNumericValue (const std::string & tag, Number & value, const std::string & prefix) const [virtual]`

6.121.3.15 `virtual bool Ipopt::OptionsList::GetIntegerValue (const std::string & tag, Index & value, const std::string & prefix) const [virtual]`

6.121.3.16 `virtual void Ipopt::OptionsList::PrintList (std::string & list) const [virtual]`

Get a string with the list of all options (tag, value, counter)

6.121.3.17 `virtual void Ipopt::OptionsList::PrintUserOptions (std::string & list) const [virtual]`

Get a string with the list of all options set by the user (tag, value, use/notused).

Here, options with dont_print flag set to true are not printed.

6.121.3.18 `virtual bool Ipopt::OptionsList::ReadFromStream (const Journalist & jnlst, std::istream & is) [virtual]`

Read options from the stream is.

Returns false if an error was encountered.

6.121.3.19 `const std::string& Ipopt::OptionsList::lowercase (const std::string tag) const [private]`

auxilliary method for converting sting to all lower-case letters

6.121.3.20 `bool Ipopt::OptionsList::find_tag (const std::string & tag, const std::string & prefix, std::string & value) const [private]`

auxilliary method for finding the value for a tag in the options list.

This method first looks for the concatenated string prefix+tag (if prefix is not ""), and if this is not found, it looks for tag. The return value is true iff prefix+tag or tag is found. In that case, the corresponding string value is copied into value.

6.121.3.21 `bool Ipopt::OptionsList::will_allow_clobber (const std::string & tag) const [private]`

tells whether or not we can clobber a particular option.

returns true if the option does not already exist, or if the option exists but is set to allow_clobber

6.121.3.22 `bool Ipopt::OptionsList::readnexttoken (std::istream & is, std::string & token) [private]`

read the next token from stream is.

Returns false, if EOF was reached before a tokens was ecountered.

6.121.4 Member Data Documentation

6.121.4.1 `std::map< std::string, OptionValue > Ipopt::OptionsList::options_` [private]

Default Constructor.

map for storing the options

Definition at line 252 of file IpOptionsList.hpp.

6.121.4.2 `SmartPtr<RegisteredOptions> Ipopt::OptionsList::reg_options_` [private]

list of all the registered options to validate against

Definition at line 255 of file IpOptionsList.hpp.

6.121.4.3 `SmartPtr<Journalist> Ipopt::OptionsList::jnlst_` [private]

[Journalist](#) for writing error messages, etc.

Definition at line 258 of file IpOptionsList.hpp.

6.121.4.4 `std::string Ipopt::OptionsList::lowercase_buffer_` [mutable],[private]

auxilliary string set by lowercase method

Definition at line 284 of file IpOptionsList.hpp.

The documentation for this class was generated from the following file:

- [Common/IpOptionsList.hpp](#)

6.122 Ipopt::OptionsList::OptionValue Class Reference

Class for storing the value and counter for each option in [OptionsList](#).

Public Member Functions

- `std::string GetValue () const`
Method for retrieving the value of an option.
- `std::string Value () const`
Method for retrieving the value without increasing the counter.
- `Index Counter () const`
Method for accessing current value of the request counter.
- `bool AllowClobber () const`
True if the option can be overwritten.
- `bool DontPrint () const`
True if this option is not to show up in the print_user_options output.

Constructors/Destructors

- `OptionValue ()`

- Default constructor (needed for the map)*
- [OptionValue](#) (std::string value, bool allow_clobber, bool dont_print)
Constructor given the value.
- [OptionValue](#) (const [OptionValue](#) ©)
Copy Constructor.
- void [operator=](#) (const [OptionValue](#) ©)
Equals operator.
- [~OptionValue](#) ()
Default Destructor.

Private Attributes

- std::string [value_](#)
Value for this option.
- [Index counter_](#)
Counter for requests.
- bool [initialized_](#)
for debugging
- bool [allow_clobber_](#)
True if the option can be overwritten.
- bool [dont_print_](#)
True if this option is not to show up in the print_user_options output.

6.122.1 Detailed Description

Class for storing the value and counter for each option in [OptionsList](#).

Definition at line 36 of file IpOptionsList.hpp.

6.122.2 Constructor & Destructor Documentation

6.122.2.1 Ipopt::OptionsList::OptionValue::OptionValue () [inline]

Default constructor (needed for the map)

Definition at line 42 of file IpOptionsList.hpp.

6.122.2.2 Ipopt::OptionsList::OptionValue::OptionValue (std::string value, bool allow_clobber, bool dont_print) [inline]

Constructor given the value.

Definition at line 48 of file IpOptionsList.hpp.

6.122.2.3 Ipopt::OptionsList::OptionValue::OptionValue (const OptionValue & copy) [inline]

Copy Constructor.

Definition at line 58 of file IpOptionsList.hpp.

6.122.2.4 Ipopt::OptionsList::OptionValue::~~OptionValue () [inline]

Default Destructor.

Definition at line 78 of file IpOptionsList.hpp.

6.122.3 Member Function Documentation

6.122.3.1 `void lpopt::OptionsList::OptionValue::operator= (const OptionValue & copy) [inline]`

Equals operator.

Definition at line 68 of file IpOptionsList.hpp.

6.122.3.2 `std::string lpopt::OptionsList::OptionValue::GetValue () const [inline]`

Method for retrieving the value of an option.

Calling this method will increase the counter by one.

Definition at line 84 of file IpOptionsList.hpp.

6.122.3.3 `std::string lpopt::OptionsList::OptionValue::Value () const [inline]`

Method for retrieving the value without increasing the counter.

Definition at line 93 of file IpOptionsList.hpp.

6.122.3.4 `Index lpopt::OptionsList::OptionValue::Counter () const [inline]`

Method for accessing current value of the request counter.

Definition at line 100 of file IpOptionsList.hpp.

6.122.3.5 `bool lpopt::OptionsList::OptionValue::AllowClobber () const [inline]`

True if the option can be overwritten.

Definition at line 107 of file IpOptionsList.hpp.

6.122.3.6 `bool lpopt::OptionsList::OptionValue::DontPrint () const [inline]`

True if this option is not to show up in the print_user_options output.

Definition at line 115 of file IpOptionsList.hpp.

6.122.4 Member Data Documentation

6.122.4.1 `std::string lpopt::OptionsList::OptionValue::value_ [private]`

Value for this option.

Definition at line 123 of file IpOptionsList.hpp.

6.122.4.2 `Index lpopt::OptionsList::OptionValue::counter_ [mutable], [private]`

Counter for requests.

Definition at line 126 of file IpOptionsList.hpp.

6.122.4.3 `bool lpopt::OptionsList::OptionValue::initialized_ [private]`

for debugging

Definition at line 129 of file IpOptionsList.hpp.

6.122.4.4 bool Ipopt::OptionsList::OptionValue::allow_clobber_ [private]

True if the option can be overwritten.

Definition at line 132 of file IpOptionsList.hpp.

6.122.4.5 bool Ipopt::OptionsList::OptionValue::dont_print_ [private]

True if this option is not to show up in the print_user_options output.

Definition at line 136 of file IpOptionsList.hpp.

The documentation for this class was generated from the following file:

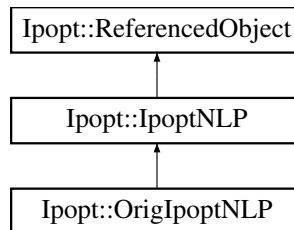
- Common/[IpOptionsList.hpp](#)

6.123 Ipopt::OrigIpoptNLP Class Reference

This class maps the traditional [NLP](#) into something that is more useful by [Ipopt](#).

```
#include <IpOrigIpoptNLP.hpp>
```

Inheritance diagram for Ipopt::OrigIpoptNLP:



Public Member Functions

- virtual bool [Initialize](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)
Initialize - overloaded from [IpoptNLP](#).
- virtual bool [InitializeStructures](#) ([SmartPtr](#)< [Vector](#) > &x, bool init_x, [SmartPtr](#)< [Vector](#) > &y_c, bool init_y_c, [SmartPtr](#)< [Vector](#) > &y_d, bool init_y_d, [SmartPtr](#)< [Vector](#) > &z_L, bool init_z_L, [SmartPtr](#)< [Vector](#) > &z_U, bool init_z_U, [SmartPtr](#)< [Vector](#) > &v_L, [SmartPtr](#)< [Vector](#) > &v_U)
Initialize (create) structures for the iteration data.
- virtual bool [GetWarmStartIterate](#) ([IteratesVector](#) &warm_start_iterate)
Method accessing the [GetWarmStartIterate](#) of the [NLP](#).
- virtual void [GetSpaces](#) ([SmartPtr](#)< const [VectorSpace](#) > &x_space, [SmartPtr](#)< const [VectorSpace](#) > &c_space, [SmartPtr](#)< const [VectorSpace](#) > &d_space, [SmartPtr](#)< const [VectorSpace](#) > &x_l_space, [SmartPtr](#)< const [MatrixSpace](#) > &px_l_space, [SmartPtr](#)< const [VectorSpace](#) > &x_u_space, [SmartPtr](#)< const [MatrixSpace](#) > &px_u_space, [SmartPtr](#)< const [VectorSpace](#) > &d_l_space, [SmartPtr](#)< const [MatrixSpace](#) > &pd_l_space, [SmartPtr](#)< const [VectorSpace](#) > &d_u_space, [SmartPtr](#)< const [MatrixSpace](#) > &pd_u_space, [SmartPtr](#)< const [MatrixSpace](#) > &Jac_c_space, [SmartPtr](#)< const [MatrixSpace](#) > &Jac_d_space, [SmartPtr](#)< const [SymMatrixSpace](#) > &Hess_lagrangian_space)
Accessor method for vector/matrix spaces pointers.
- virtual void [AdjustVariableBounds](#) (const [Vector](#) &new_x_L, const [Vector](#) &new_x_U, const [Vector](#) &new_d_L, const [Vector](#) &new_d_U)
Method for adapting the variable bounds.

- `SmartPtr< NLP > nlp ()`

Accessor method to the underlying NLP.

Constructors/Destructors

- `OriglpoptNLP (const SmartPtr< const Journalist > &jnlst, const SmartPtr< NLP > &nlp, const SmartPtr< NLPScalingObject > &nlp_scaling)`
- `virtual ~OriglpoptNLP ()`

Default destructor.

- `virtual Number f (const Vector &x)`

Accessor methods for model data.

- `virtual Number f (const Vector &x, Number mu)`

Objective value (depending in mu) - incorrect version for OriglpoptNLP.

- `virtual SmartPtr< const Vector > grad_f (const Vector &x)`

Gradient of the objective.

- `virtual SmartPtr< const Vector > grad_f (const Vector &x, Number mu)`

Gradient of the objective (depending in mu) - incorrect version for OriglpoptNLP.

- `virtual SmartPtr< const Vector > c (const Vector &x)`

Equality constraint residual.

- `virtual SmartPtr< const Matrix > jac_c (const Vector &x)`

Jacobian Matrix for equality constraints.

- `virtual SmartPtr< const Vector > d (const Vector &x)`

Inequality constraint residual (reformulated as equalities with slacks).

- `virtual SmartPtr< const Matrix > jac_d (const Vector &x)`

Jacobian Matrix for inequality constraints.

- `virtual SmartPtr< const SymMatrix > h (const Vector &x, Number obj_factor, const Vector &yc, const Vector &y)`

Hessian of the Lagrangian.

- `virtual SmartPtr< const SymMatrix > h (const Vector &x, Number obj_factor, const Vector &yc, const Vector &y, Number mu)`

Hessian of the Lagrangian (depending in mu) - incorrect version for OriglpoptNLP.

- `virtual SmartPtr< const SymMatrix > uninitialized_h ()`

Provides a Hessian matrix from the correct matrix space with uninitialized values.

- `virtual SmartPtr< const Vector > x_L () const`

Lower bounds on x.

- `virtual SmartPtr< const Matrix > Px_L () const`

Permutation matrix (x_L -> x)

- `virtual SmartPtr< const Vector > x_U () const`

Upper bounds on x.

- `virtual SmartPtr< const Matrix > Px_U () const`

Permutation matrix (x_U -> x).

- `virtual SmartPtr< const Vector > d_L () const`

Lower bounds on d.

- `virtual SmartPtr< const Matrix > Pd_L () const`

Permutation matrix (d_L -> d)

- `virtual SmartPtr< const Vector > d_U () const`

Upper bounds on d.

- `virtual SmartPtr< const Matrix > Pd_U () const`

Permutation matrix (d_U -> d).

- virtual [SmartPtr](#)< const [SymMatrixSpace](#) > [HessianMatrixSpace](#) () const

Accessor method to obtain the [MatrixSpace](#) for the Hessian matrix (or it's approximation)

Counters for the number of function evaluations.

- virtual [Index](#) [f_evals](#) () const
- virtual [Index](#) [grad_f_evals](#) () const
- virtual [Index](#) [c_evals](#) () const
- virtual [Index](#) [jac_c_evals](#) () const
- virtual [Index](#) [d_evals](#) () const
- virtual [Index](#) [jac_d_evals](#) () const
- virtual [Index](#) [h_evals](#) () const
- void [FinalizeSolution](#) ([SolverReturn](#) status, const [Vector](#) &x, const [Vector](#) &z_L, const [Vector](#) &z_U, const [Vector](#) &c, const [Vector](#) &d, const [Vector](#) &y_c, const [Vector](#) &y_d, [Number](#) obj_value, const [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)

Solution Routines - overloaded from [IpoptNLP](#).

- bool [IntermediateCallback](#) ([AlgorithmMode](#) mode, [Index](#) iter, [Number](#) obj_value, [Number](#) inf_pr, [Number](#) inf_du, [Number](#) mu, [Number](#) d_norm, [Number](#) regularization_size, [Number](#) alpha_du, [Number](#) alpha_pr, [Index](#) ls_trials, [SmartPtr](#)< const [IpoptData](#) > ip_data, [SmartPtr](#)< [IpoptCalculatedQuantities](#) > ip_cq)

Methods related to function evaluation timing.

- void [ResetTimes](#) ()
Reset the timing statistics.
- void [PrintTimingStatistics](#) ([Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category) const
- const [TimedTask](#) & [f_eval_time](#) () const
- const [TimedTask](#) & [grad_f_eval_time](#) () const
- const [TimedTask](#) & [c_eval_time](#) () const
- const [TimedTask](#) & [jac_c_eval_time](#) () const
- const [TimedTask](#) & [d_eval_time](#) () const
- const [TimedTask](#) & [jac_d_eval_time](#) () const
- const [TimedTask](#) & [h_eval_time](#) () const
- [Number](#) [TotalFunctionEvaluationCpuTime](#) () const
- [Number](#) [TotalFunctionEvaluationSysTime](#) () const
- [Number](#) [TotalFunctionEvaluationWallclockTime](#) () const

Static Public Member Functions

Methods for IpoptType

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Called by [IpoptType](#) to register the options.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `OriglpoptNLP ()`
Default Constructor.
- `OriglpoptNLP (const OriglpoptNLP &)`
Copy Constructor.
- `void operator= (const OriglpoptNLP &)`
Overloaded Equals Operator.

auxilliary functions

- `void relax_bounds (Number bound_relax_factor, Vector &bounds)`
relax the bounds by a relative move of relax_bound_factor.
- `SmartPtr< const Vector > get_unscaled_x (const Vector &x)`
Method for getting the unscaled version of the x vector.

Private Attributes

- `SmartPtr< const Journalist > jnlst_`
journalist
- `SmartPtr< NLP > nlp_`
Pointer to the NLP.
- `bool initialized_`
Flag indicating if initialization method has been called.
- `SmartPtr< const VectorSpace > x_space_`
Necessary Vector/Matrix spaces.
- `SmartPtr< const VectorSpace > c_space_`
- `SmartPtr< const VectorSpace > d_space_`
- `SmartPtr< const VectorSpace > x_l_space_`
- `SmartPtr< const MatrixSpace > px_l_space_`
- `SmartPtr< const VectorSpace > x_u_space_`
- `SmartPtr< const MatrixSpace > px_u_space_`
- `SmartPtr< const VectorSpace > d_l_space_`
- `SmartPtr< const MatrixSpace > pd_l_space_`
- `SmartPtr< const VectorSpace > d_u_space_`
- `SmartPtr< const MatrixSpace > pd_u_space_`
- `SmartPtr< const MatrixSpace > jac_c_space_`
- `SmartPtr< const MatrixSpace > jac_d_space_`
- `SmartPtr< const SymMatrixSpace > h_space_`
- `SmartPtr< const MatrixSpace > scaled_jac_c_space_`
- `SmartPtr< const MatrixSpace > scaled_jac_d_space_`
- `SmartPtr< const SymMatrixSpace > scaled_h_space_`

Storage for Model Quantities

- `CachedResults< Number > f_cache_`
Objective function.
- `CachedResults< SmartPtr< const Vector > > grad_f_cache_`
Gradient of the objective function.
- `CachedResults< SmartPtr< const Vector > > c_cache_`

- Equality constraint residuals.
 - [CachedResults](#)< [SmartPtr](#)< const [Matrix](#) > > [jac_c_cache_](#)
Jacobian [Matrix](#) for equality constraints (current iteration)
 - [CachedResults](#)< [SmartPtr](#)< const [Vector](#) > > [d_cache_](#)
Inequality constraint residual (reformulated as equalities with slacks).
 - [CachedResults](#)< [SmartPtr](#)< const [Matrix](#) > > [jac_d_cache_](#)
Jacobian [Matrix](#) for inequality constraints (current iteration)
 - [CachedResults](#)< [SmartPtr](#)< const [SymMatrix](#) > > [h_cache_](#)
Hessian of the lagrangian (current iteration)
 - [CachedResults](#)< [SmartPtr](#)< const [Vector](#) > > [unscaled_x_cache_](#)
Unscaled version of x vector.
 - [SmartPtr](#)< const [Vector](#) > [x_L_](#)
Lower bounds on x.
 - [SmartPtr](#)< const [Matrix](#) > [Px_L_](#)
Permutation matrix ($x_L \rightarrow x$)
 - [SmartPtr](#)< const [Vector](#) > [x_U_](#)
Upper bounds on x.
 - [SmartPtr](#)< const [Matrix](#) > [Px_U_](#)
Permutation matrix ($x_U \rightarrow x$).
 - [SmartPtr](#)< const [Vector](#) > [d_L_](#)
Lower bounds on d.
 - [SmartPtr](#)< const [Matrix](#) > [Pd_L_](#)
Permutation matrix ($d_L \rightarrow d$)
 - [SmartPtr](#)< const [Vector](#) > [d_U_](#)
Upper bounds on d.
 - [SmartPtr](#)< const [Matrix](#) > [Pd_U_](#)
Permutation matrix ($d_U \rightarrow d$).
 - [SmartPtr](#)< const [Vector](#) > [orig_x_L_](#)
Original unmodified lower bounds on x.
 - [SmartPtr](#)< const [Vector](#) > [orig_x_U_](#)
Original unmodified upper bounds on x.

Algorithmic parameters

- [Number bound_relax_factor_](#)
relaxation factor for the bounds
- bool [honor_original_bounds_](#)
Flag indicating whether the primal variables should be projected back into original bounds are optimization.
- bool [warm_start_same_structure_](#)
Flag indicating whether the *TNLP* with identical structure has already been solved before.
- [HessianApproximationType](#) [hessian_approximation_](#)
Flag indicating what Hessian information is to be used.
- [HessianApproximationSpace](#) [hessian_approximation_space_](#)
Flag indicating in which space Hessian is to be approximated.
- bool [check_derivatives_for_naninf_](#)
Flag indicating whether it is desired to check if there are Nan or Inf entries in first and second derivative matrices.
- bool [jac_c_constant_](#)
Flag indicating if we need to ask for equality constraint Jacobians only once.
- bool [jac_d_constant_](#)
Flag indicating if we need to ask for inequality constraint Jacobians only once.

- bool [hessian_constant_](#)
Flag indicating if we need to ask for Hessian only once.

Counters for the function evaluations

- [Index f_evals_](#)
- [Index grad_f_evals_](#)
- [Index c_evals_](#)
- [Index jac_c_evals_](#)
- [Index d_evals_](#)
- [Index jac_d_evals_](#)
- [Index h_evals_](#)

Timing statistics for the function evaluations.

- [TimedTask f_eval_time_](#)
- [TimedTask grad_f_eval_time_](#)
- [TimedTask c_eval_time_](#)
- [TimedTask jac_c_eval_time_](#)
- [TimedTask d_eval_time_](#)
- [TimedTask jac_d_eval_time_](#)
- [TimedTask h_eval_time_](#)

6.123.1 Detailed Description

This class maps the traditional [NLP](#) into something that is more useful by [lpopt](#).

This class takes care of storing the calculated model results, handles caching, and (some day) takes care of addition of slacks.

Definition at line 37 of file `lpOriglpoptNLP.hpp`.

6.123.2 Constructor & Destructor Documentation

6.123.2.1 `lpopt::OriglpoptNLP::OriglpoptNLP (const SmartPtr< const Journalist > & jnlst, const SmartPtr< NLP > & nlp, const SmartPtr< NLPScalingObject > & nlp_scaling)`

6.123.2.2 `virtual lpopt::OriglpoptNLP::~OriglpoptNLP () [virtual]`

Default destructor.

6.123.2.3 `lpopt::OriglpoptNLP::OriglpoptNLP () [private]`

Default Constructor.

6.123.2.4 `lpopt::OriglpoptNLP::OriglpoptNLP (const OriglpoptNLP &) [private]`

Copy Constructor.

6.123.3 Member Function Documentation

6.123.3.1 `virtual bool lpopt::OriglpoptNLP::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [virtual]`

Initialize - overloaded from [lpoptNLP](#).

Reimplemented from [lpopt::lpoptNLP](#).

6.123.3.2 `virtual bool Ipopt::OrigIpoptNLP::InitializeStructures (SmartPtr< Vector > & x, bool init_x, SmartPtr< Vector > & y_c, bool init_y_c, SmartPtr< Vector > & y_d, bool init_y_d, SmartPtr< Vector > & z_L, bool init_z_L, SmartPtr< Vector > & z_U, bool init_z_U, SmartPtr< Vector > & v_L, SmartPtr< Vector > & v_U)`
`[virtual]`

Initialize (create) structures for the iteration data.

Implements [Ipopt::IpoptNLP](#).

6.123.3.3 `virtual bool Ipopt::OrigIpoptNLP::GetWarmStartIterate (IteratesVector & warm_start_iterate)` `[inline]`,
`[virtual]`

Method accessing the GetWarmStartIterate of the [NLP](#).

Implements [Ipopt::IpoptNLP](#).

Definition at line 72 of file IpOrigIpoptNLP.hpp.

6.123.3.4 `virtual Number Ipopt::OrigIpoptNLP::f (const Vector & x)` `[virtual]`

Accessor methods for model data.

Objective value

Implements [Ipopt::IpoptNLP](#).

6.123.3.5 `virtual Number Ipopt::OrigIpoptNLP::f (const Vector & x, Number mu)` `[virtual]`

Objective value (depending in mu) - incorrect version for [OrigIpoptNLP](#).

Implements [Ipopt::IpoptNLP](#).

6.123.3.6 `virtual SmartPtr<const Vector> Ipopt::OrigIpoptNLP::grad_f (const Vector & x)` `[virtual]`

Gradient of the objective.

Implements [Ipopt::IpoptNLP](#).

6.123.3.7 `virtual SmartPtr<const Vector> Ipopt::OrigIpoptNLP::grad_f (const Vector & x, Number mu)` `[virtual]`

Gradient of the objective (depending in mu) - incorrect version for [OrigIpoptNLP](#).

Implements [Ipopt::IpoptNLP](#).

6.123.3.8 `virtual SmartPtr<const Vector> Ipopt::OrigIpoptNLP::c (const Vector & x)` `[virtual]`

Equality constraint residual.

Implements [Ipopt::IpoptNLP](#).

6.123.3.9 `virtual SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::jac_c (const Vector & x)` `[virtual]`

Jacobian [Matrix](#) for equality constraints.

Implements [Ipopt::IpoptNLP](#).

6.123.3.10 `virtual SmartPtr<const Vector> Ipopt::OrigIpoptNLP::d (const Vector & x)` `[virtual]`

Inequality constraint residual (reformulated as equalities with slacks).

Implements [Ipopt::IpoptNLP](#).

6.123.3.11 `virtual SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::jac_d (const Vector & x) [virtual]`

Jacobian [Matrix](#) for inequality constraints.

Implements [Ipopt::IpoptNLP](#).

6.123.3.12 `virtual SmartPtr<const SymMatrix> Ipopt::OrigIpoptNLP::h (const Vector & x, Number obj_factor, const Vector & yc, const Vector & yd) [virtual]`

Hessian of the Lagrangian.

Implements [Ipopt::IpoptNLP](#).

6.123.3.13 `virtual SmartPtr<const SymMatrix> Ipopt::OrigIpoptNLP::h (const Vector & x, Number obj_factor, const Vector & yc, const Vector & yd, Number mu) [virtual]`

Hessian of the Lagrangian (depending in mu) - incorrect version for [OrigIpoptNLP](#).

Implements [Ipopt::IpoptNLP](#).

6.123.3.14 `virtual SmartPtr<const SymMatrix> Ipopt::OrigIpoptNLP::uninitialized_h () [virtual]`

Provides a Hessian matrix from the correct matrix space with uninitialized values.

This can be used in LeastSquareMults to obtain a "zero Hessian".

Implements [Ipopt::IpoptNLP](#).

6.123.3.15 `virtual SmartPtr<const Vector> Ipopt::OrigIpoptNLP::x_L () const [inline],[virtual]`

Lower bounds on x.

Implements [Ipopt::IpoptNLP](#).

Definition at line 126 of file IpOrigIpoptNLP.hpp.

6.123.3.16 `virtual SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::Px_L () const [inline],[virtual]`

Permutation matrix (x_L_ -> x)

Implements [Ipopt::IpoptNLP](#).

Definition at line 132 of file IpOrigIpoptNLP.hpp.

6.123.3.17 `virtual SmartPtr<const Vector> Ipopt::OrigIpoptNLP::x_U () const [inline],[virtual]`

Upper bounds on x.

Implements [Ipopt::IpoptNLP](#).

Definition at line 138 of file IpOrigIpoptNLP.hpp.

6.123.3.18 `virtual SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::Px_U () const [inline],[virtual]`

Permutation matrix (x_U_ -> x).

Implements [Ipopt::IpoptNLP](#).

Definition at line 144 of file IpOrigIpoptNLP.hpp.

6.123.3.19 `virtual SmartPtr<const Vector> Ipopt::OrigIpoptNLP::d_L () const [inline],[virtual]`

Lower bounds on d.

Implements [Ipopt::IpoptNLP](#).

Definition at line 150 of file IpOrigIpoptNLP.hpp.

6.123.3.20 `virtual SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::Pd_L () const [inline],[virtual]`

Permutation matrix (d_L_ -> d)

Implements [Ipopt::IpoptNLP](#).

Definition at line 156 of file IpOrigIpoptNLP.hpp.

6.123.3.21 `virtual SmartPtr<const Vector> Ipopt::OrigIpoptNLP::d_U () const [inline],[virtual]`

Upper bounds on d.

Implements [Ipopt::IpoptNLP](#).

Definition at line 162 of file IpOrigIpoptNLP.hpp.

6.123.3.22 `virtual SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::Pd_U () const [inline],[virtual]`

Permutation matrix (d_U_ -> d.

Implements [Ipopt::IpoptNLP](#).

Definition at line 168 of file IpOrigIpoptNLP.hpp.

6.123.3.23 `virtual SmartPtr<const SymMatrixSpace> Ipopt::OrigIpoptNLP::HessianMatrixSpace () const [inline],[virtual]`

Accessor method to obtain the [MatrixSpace](#) for the Hessian matrix (or it's approximation)

Implements [Ipopt::IpoptNLP](#).

Definition at line 173 of file IpOrigIpoptNLP.hpp.

6.123.3.24 `virtual void Ipopt::OrigIpoptNLP::GetSpaces (SmartPtr< const VectorSpace > & x_space, SmartPtr< const VectorSpace > & c_space, SmartPtr< const VectorSpace > & d_space, SmartPtr< const VectorSpace > & x_l_space, SmartPtr< const MatrixSpace > & px_l_space, SmartPtr< const VectorSpace > & x_u_space, SmartPtr< const MatrixSpace > & px_u_space, SmartPtr< const VectorSpace > & d_l_space, SmartPtr< const MatrixSpace > & pd_l_space, SmartPtr< const VectorSpace > & d_u_space, SmartPtr< const MatrixSpace > & pd_u_space, SmartPtr< const MatrixSpace > & Jac_c_space, SmartPtr< const MatrixSpace > & Jac_d_space, SmartPtr< const SymMatrixSpace > & Hess_lagrangian_space) [virtual]`

Accessor method for vector/matrix spaces pointers.

Implements [Ipopt::IpoptNLP](#).

6.123.3.25 `virtual void Ipopt::OrigIpoptNLP::AdjustVariableBounds (const Vector & new_x_L, const Vector & new_x_U, const Vector & new_d_L, const Vector & new_d_U) [virtual]`

Method for adapting the variable bounds.

This is called if slacks are becoming too small

Implements [Ipopt::IpoptNLP](#).

6.123.3.26 `virtual Index Ipopt::OrigIpoptNLP::f_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 204 of file IpOrigIpoptNLP.hpp.

6.123.3.27 `virtual Index Ipopt::OrigIpoptNLP::grad_f_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 208 of file IpOrigIpoptNLP.hpp.

6.123.3.28 `virtual Index Ipopt::OrigIpoptNLP::c_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 212 of file IpOrigIpoptNLP.hpp.

6.123.3.29 `virtual Index Ipopt::OrigIpoptNLP::jac_c_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 216 of file IpOrigIpoptNLP.hpp.

6.123.3.30 `virtual Index Ipopt::OrigIpoptNLP::d_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 220 of file IpOrigIpoptNLP.hpp.

6.123.3.31 `virtual Index Ipopt::OrigIpoptNLP::jac_d_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 224 of file IpOrigIpoptNLP.hpp.

6.123.3.32 `virtual Index Ipopt::OrigIpoptNLP::h_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 228 of file IpOrigIpoptNLP.hpp.

6.123.3.33 `void Ipopt::OrigIpoptNLP::FinalizeSolution (SolverReturn status, const Vector & x, const Vector & z_L, const Vector & z_U, const Vector & c, const Vector & d, const Vector & y_c, const Vector & y_d, Number obj_value, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [virtual]`

Solution Routines - overloaded from [IpoptNLP](#).

Implements [Ipopt::IpoptNLP](#).

6.123.3.34 `bool Ipopt::OrigIpoptNLP::IntermediateCallBack (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, SmartPtr< const IpoptData > ip_data, SmartPtr< IpoptCalculatedQuantities > ip_cq) [virtual]`

Implements [Ipopt::IpoptNLP](#).

6.123.3.35 `static void Ipopt::OrigIpoptNLP::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Called by IpoptType to register the options.

6.123.3.36 `SmartPtr<NLP> Ipopt::OrigIpoptNLP::nlp () [inline]`

Accessor method to the underlying [NLP](#).

Definition at line 261 of file IpOrigIpoptNLP.hpp.

6.123.3.37 void Ipopt::OrigIpoptNLP::ResetTimes ()

Reset the timing statistics.

6.123.3.38 void Ipopt::OrigIpoptNLP::PrintTimingStatistics (**Journalist & *jnlst***, **EJournalLevel *level***, **EJournalCategory *category***) const

6.123.3.39 const TimedTask& Ipopt::OrigIpoptNLP::f_eval_time () const [inline]

Definition at line 276 of file IpOrigIpoptNLP.hpp.

6.123.3.40 const TimedTask& Ipopt::OrigIpoptNLP::grad_f_eval_time () const [inline]

Definition at line 280 of file IpOrigIpoptNLP.hpp.

6.123.3.41 const TimedTask& Ipopt::OrigIpoptNLP::c_eval_time () const [inline]

Definition at line 284 of file IpOrigIpoptNLP.hpp.

6.123.3.42 const TimedTask& Ipopt::OrigIpoptNLP::jac_c_eval_time () const [inline]

Definition at line 288 of file IpOrigIpoptNLP.hpp.

6.123.3.43 const TimedTask& Ipopt::OrigIpoptNLP::d_eval_time () const [inline]

Definition at line 292 of file IpOrigIpoptNLP.hpp.

6.123.3.44 const TimedTask& Ipopt::OrigIpoptNLP::jac_d_eval_time () const [inline]

Definition at line 296 of file IpOrigIpoptNLP.hpp.

6.123.3.45 const TimedTask& Ipopt::OrigIpoptNLP::h_eval_time () const [inline]

Definition at line 300 of file IpOrigIpoptNLP.hpp.

6.123.3.46 Number Ipopt::OrigIpoptNLP::TotalFunctionEvaluationCpuTime () const

6.123.3.47 Number Ipopt::OrigIpoptNLP::TotalFunctionEvaluationSysTime () const

6.123.3.48 Number Ipopt::OrigIpoptNLP::TotalFunctionEvaluationWallclockTime () const

6.123.3.49 void Ipopt::OrigIpoptNLP::operator= (const OrigIpoptNLP &) [private]

Overloaded Equals Operator.

6.123.3.50 void Ipopt::OrigIpoptNLP::relax_bounds (**Number *bound_relax_factor***, **Vector & *bounds***) [private]

relax the bounds by a relative move of `relax_bound_factor`.

Here, `relax_bound_factor` should be negative (or zero) for lower bounds, and positive (or zero) for upper bounds.

6.123.3.51 SmartPtr<const Vector> Ipopt::OrigIpoptNLP::get_unscaled_x (const Vector & x) [private]

Method for getting the unscaled version of the x vector.

6.123.4 Member Data Documentation

6.123.4.1 **SmartPtr<const Journalist>** Ipopt::OriglpoptNLP::jnlst_ [private]

journalist

Definition at line 312 of file IpOriglpoptNLP.hpp.

6.123.4.2 **SmartPtr<NLP>** Ipopt::OriglpoptNLP::nlp_ [private]

Pointer to the [NLP](#).

Definition at line 315 of file IpOriglpoptNLP.hpp.

6.123.4.3 **SmartPtr<const VectorSpace>** Ipopt::OriglpoptNLP::x_space_ [private]

Necessary Vector/Matrix spaces.

Definition at line 319 of file IpOriglpoptNLP.hpp.

6.123.4.4 **SmartPtr<const VectorSpace>** Ipopt::OriglpoptNLP::c_space_ [private]

Definition at line 320 of file IpOriglpoptNLP.hpp.

6.123.4.5 **SmartPtr<const VectorSpace>** Ipopt::OriglpoptNLP::d_space_ [private]

Definition at line 321 of file IpOriglpoptNLP.hpp.

6.123.4.6 **SmartPtr<const VectorSpace>** Ipopt::OriglpoptNLP::x_l_space_ [private]

Definition at line 322 of file IpOriglpoptNLP.hpp.

6.123.4.7 **SmartPtr<const MatrixSpace>** Ipopt::OriglpoptNLP::px_l_space_ [private]

Definition at line 323 of file IpOriglpoptNLP.hpp.

6.123.4.8 **SmartPtr<const VectorSpace>** Ipopt::OriglpoptNLP::x_u_space_ [private]

Definition at line 324 of file IpOriglpoptNLP.hpp.

6.123.4.9 **SmartPtr<const MatrixSpace>** Ipopt::OriglpoptNLP::px_u_space_ [private]

Definition at line 325 of file IpOriglpoptNLP.hpp.

6.123.4.10 **SmartPtr<const VectorSpace>** Ipopt::OriglpoptNLP::d_l_space_ [private]

Definition at line 326 of file IpOriglpoptNLP.hpp.

6.123.4.11 **SmartPtr<const MatrixSpace>** Ipopt::OriglpoptNLP::pd_l_space_ [private]

Definition at line 327 of file IpOriglpoptNLP.hpp.

6.123.4.12 **SmartPtr<const VectorSpace>** Ipopt::OriglpoptNLP::d_u_space_ [private]

Definition at line 328 of file IpOriglpoptNLP.hpp.

6.123.4.13 **SmartPtr<const MatrixSpace>** Ipopt::OriglpoptNLP::pd_u_space_ [private]

Definition at line 329 of file IpOriglpoptNLP.hpp.

6.123.4.14 **SmartPtr<const MatrixSpace>** Ipopt::OrigIpoptNLP::jac_c_space_ [private]

Definition at line 330 of file IpOrigIpoptNLP.hpp.

6.123.4.15 **SmartPtr<const MatrixSpace>** Ipopt::OrigIpoptNLP::jac_d_space_ [private]

Definition at line 331 of file IpOrigIpoptNLP.hpp.

6.123.4.16 **SmartPtr<const SymMatrixSpace>** Ipopt::OrigIpoptNLP::h_space_ [private]

Definition at line 332 of file IpOrigIpoptNLP.hpp.

6.123.4.17 **SmartPtr<const MatrixSpace>** Ipopt::OrigIpoptNLP::scaled_jac_c_space_ [private]

Definition at line 334 of file IpOrigIpoptNLP.hpp.

6.123.4.18 **SmartPtr<const MatrixSpace>** Ipopt::OrigIpoptNLP::scaled_jac_d_space_ [private]

Definition at line 335 of file IpOrigIpoptNLP.hpp.

6.123.4.19 **SmartPtr<const SymMatrixSpace>** Ipopt::OrigIpoptNLP::scaled_h_space_ [private]

Definition at line 336 of file IpOrigIpoptNLP.hpp.

6.123.4.20 **CachedResults<Number>** Ipopt::OrigIpoptNLP::f_cache_ [private]

Objective function.

Definition at line 341 of file IpOrigIpoptNLP.hpp.

6.123.4.21 **CachedResults<SmartPtr<const Vector> >** Ipopt::OrigIpoptNLP::grad_f_cache_ [private]

Gradient of the objective function.

Definition at line 344 of file IpOrigIpoptNLP.hpp.

6.123.4.22 **CachedResults<SmartPtr<const Vector> >** Ipopt::OrigIpoptNLP::c_cache_ [private]

Equality constraint residuals.

Definition at line 347 of file IpOrigIpoptNLP.hpp.

6.123.4.23 **CachedResults<SmartPtr<const Matrix> >** Ipopt::OrigIpoptNLP::jac_c_cache_ [private]

Jacobian [Matrix](#) for equality constraints (current iteration)

Definition at line 351 of file IpOrigIpoptNLP.hpp.

6.123.4.24 **CachedResults<SmartPtr<const Vector> >** Ipopt::OrigIpoptNLP::d_cache_ [private]

Inequality constraint residual (reformulated as equalities with slacks).

Definition at line 355 of file IpOrigIpoptNLP.hpp.

6.123.4.25 **CachedResults<SmartPtr<const Matrix> >** Ipopt::OrigIpoptNLP::jac_d_cache_ [private]

Jacobian [Matrix](#) for inequality constraints (current iteration)

Definition at line 359 of file IpOrigIpoptNLP.hpp.

6.123.4.26 `CachedResults<SmartPtr<const SymMatrix>> Ipopt::OrigIpoptNLP::h_cache_` [private]

Hessian of the lagrangian (current iteration)

Definition at line 363 of file IpOrigIpoptNLP.hpp.

6.123.4.27 `CachedResults<SmartPtr<const Vector>> Ipopt::OrigIpoptNLP::unscaled_x_cache_` [private]

Unscaled version of x vector.

Definition at line 366 of file IpOrigIpoptNLP.hpp.

6.123.4.28 `SmartPtr<const Vector> Ipopt::OrigIpoptNLP::x_L_` [private]

Lower bounds on x.

Definition at line 369 of file IpOrigIpoptNLP.hpp.

6.123.4.29 `SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::Px_L_` [private]

Permutation matrix ($x_L_ \rightarrow x$)

Definition at line 372 of file IpOrigIpoptNLP.hpp.

6.123.4.30 `SmartPtr<const Vector> Ipopt::OrigIpoptNLP::x_U_` [private]

Upper bounds on x.

Definition at line 375 of file IpOrigIpoptNLP.hpp.

6.123.4.31 `SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::Px_U_` [private]

Permutation matrix ($x_U_ \rightarrow x$).

Definition at line 378 of file IpOrigIpoptNLP.hpp.

6.123.4.32 `SmartPtr<const Vector> Ipopt::OrigIpoptNLP::d_L_` [private]

Lower bounds on d.

Definition at line 381 of file IpOrigIpoptNLP.hpp.

6.123.4.33 `SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::Pd_L_` [private]

Permutation matrix ($d_L_ \rightarrow d$)

Definition at line 384 of file IpOrigIpoptNLP.hpp.

6.123.4.34 `SmartPtr<const Vector> Ipopt::OrigIpoptNLP::d_U_` [private]

Upper bounds on d.

Definition at line 387 of file IpOrigIpoptNLP.hpp.

6.123.4.35 `SmartPtr<const Matrix> Ipopt::OrigIpoptNLP::Pd_U_` [private]

Permutation matrix ($d_U_ \rightarrow d$).

Definition at line 390 of file IpOrigIpoptNLP.hpp.

6.123.4.36 `SmartPtr<const Vector> Ipopt::OrigIpoptNLP::orig_x_L_ [private]`

Original unmodified lower bounds on x.

Definition at line 393 of file IpOrigIpoptNLP.hpp.

6.123.4.37 `SmartPtr<const Vector> Ipopt::OrigIpoptNLP::orig_x_U_ [private]`

Original unmodified upper bounds on x.

Definition at line 396 of file IpOrigIpoptNLP.hpp.

6.123.4.38 `Number Ipopt::OrigIpoptNLP::bound_relax_factor_ [private]`

relaxation factor for the bounds

Definition at line 431 of file IpOrigIpoptNLP.hpp.

6.123.4.39 `bool Ipopt::OrigIpoptNLP::honor_original_bounds_ [private]`

Flag indicating whether the primal variables should be projected back into original bounds are optimization.

Definition at line 434 of file IpOrigIpoptNLP.hpp.

6.123.4.40 `bool Ipopt::OrigIpoptNLP::warm_start_same_structure_ [private]`

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 437 of file IpOrigIpoptNLP.hpp.

6.123.4.41 `HessianApproximationType Ipopt::OrigIpoptNLP::hessian_approximation_ [private]`

Flag indicating what Hessian information is to be used.

Definition at line 439 of file IpOrigIpoptNLP.hpp.

6.123.4.42 `HessianApproximationSpace Ipopt::OrigIpoptNLP::hessian_approximation_space_ [private]`

Flag indicating in which space Hessian is to be approximated.

Definition at line 441 of file IpOrigIpoptNLP.hpp.

6.123.4.43 `bool Ipopt::OrigIpoptNLP::check_derivatives_for_naninf_ [private]`

Flag indicating whether it is desired to check if there are Nan or Inf entries in first and second derivative matrices.

Definition at line 444 of file IpOrigIpoptNLP.hpp.

6.123.4.44 `bool Ipopt::OrigIpoptNLP::jac_c_constant_ [private]`

Flag indicating if we need to ask for equality constraint Jacobians only once.

Definition at line 447 of file IpOrigIpoptNLP.hpp.

6.123.4.45 `bool Ipopt::OrigIpoptNLP::jac_d_constant_ [private]`

Flag indicating if we need to ask for inequality constraint Jacobians only once.

Definition at line 450 of file IpOrigIpoptNLP.hpp.

6.123.4.46 `bool Ipopt::OrigIpoptNLP::hessian_constant_ [private]`

Flag indicating if we need to ask for Hessian only once.

Definition at line 452 of file IpOrigIpoptNLP.hpp.

6.123.4.47 `Index Ipopt::OrigIpoptNLP::f_evals_ [private]`

Definition at line 457 of file IpOrigIpoptNLP.hpp.

6.123.4.48 `Index Ipopt::OrigIpoptNLP::grad_f_evals_ [private]`

Definition at line 458 of file IpOrigIpoptNLP.hpp.

6.123.4.49 `Index Ipopt::OrigIpoptNLP::c_evals_ [private]`

Definition at line 459 of file IpOrigIpoptNLP.hpp.

6.123.4.50 `Index Ipopt::OrigIpoptNLP::jac_c_evals_ [private]`

Definition at line 460 of file IpOrigIpoptNLP.hpp.

6.123.4.51 `Index Ipopt::OrigIpoptNLP::d_evals_ [private]`

Definition at line 461 of file IpOrigIpoptNLP.hpp.

6.123.4.52 `Index Ipopt::OrigIpoptNLP::jac_d_evals_ [private]`

Definition at line 462 of file IpOrigIpoptNLP.hpp.

6.123.4.53 `Index Ipopt::OrigIpoptNLP::h_evals_ [private]`

Definition at line 463 of file IpOrigIpoptNLP.hpp.

6.123.4.54 `bool Ipopt::OrigIpoptNLP::initialized_ [private]`

Flag indicating if initialization method has been called.

Definition at line 467 of file IpOrigIpoptNLP.hpp.

6.123.4.55 `TimedTask Ipopt::OrigIpoptNLP::f_eval_time_ [private]`

Definition at line 471 of file IpOrigIpoptNLP.hpp.

6.123.4.56 `TimedTask Ipopt::OrigIpoptNLP::grad_f_eval_time_ [private]`

Definition at line 472 of file IpOrigIpoptNLP.hpp.

6.123.4.57 `TimedTask Ipopt::OrigIpoptNLP::c_eval_time_ [private]`

Definition at line 473 of file IpOrigIpoptNLP.hpp.

6.123.4.58 `TimedTask Ipopt::OrigIpoptNLP::jac_c_eval_time_ [private]`

Definition at line 474 of file IpOrigIpoptNLP.hpp.

6.123.4.59 `TimedTask Ipopt::OrigIpoptNLP::d_eval_time_ [private]`

Definition at line 475 of file IpOrigIpoptNLP.hpp.

6.123.4.60 TimedTask Ipopt::OrigIpoptNLP::jac_d_eval_time_ [private]

Definition at line 476 of file IpOrigIpoptNLP.hpp.

6.123.4.61 TimedTask Ipopt::OrigIpoptNLP::h_eval_time_ [private]

Definition at line 477 of file IpOrigIpoptNLP.hpp.

The documentation for this class was generated from the following file:

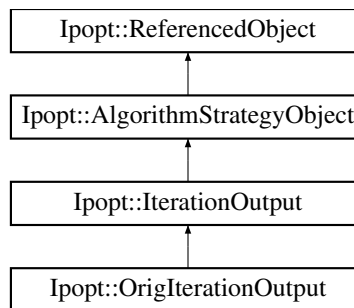
- Algorithm/IpOrigIpoptNLP.hpp

6.124 Ipopt::OrigIterationOutput Class Reference

Class for the iteration summary output for the original [NLP](#).

```
#include <IpOrigIterationOutput.hpp>
```

Inheritance diagram for Ipopt::OrigIterationOutput:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual void [WriteOutput](#) ()
Method to do all the summary output per iteration.

Constructors/Destructors

- [OrigIterationOutput](#) ()
Default Constructor.
- virtual [~OrigIterationOutput](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (SmartPtr< [RegisteredOptions](#) > roptions)
Methods for [OptionsList](#).

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [OrigIterationOutput](#) (const [OrigIterationOutput](#) &)
Copy Constructor.
- void [operator=](#) (const [OrigIterationOutput](#) &)
Overloaded Equals Operator.

Private Attributes

- bool [print_info_string_](#)
Flag indicating weather info string should be printed at end of iteration summary line.
- [InfPrOutput](#) [inf_pr_output_](#)
Option indication what should be printed in inf_pr column.
- int [print_frequency_iter_](#)
Option indicating at which iteration frequency the summary line should be printed.
- [Number](#) [print_frequency_time_](#)
Option indicating at which time frequency the summary line should be printed.

Additional Inherited Members

6.124.1 Detailed Description

Class for the iteration summary output for the original [NLP](#).

Definition at line 19 of file [IpOrigIterationOutput.hpp](#).

6.124.2 Constructor & Destructor Documentation

6.124.2.1 [Ipopt::OrigIterationOutput::OrigIterationOutput](#) ()

Default Constructor.

6.124.2.2 [virtual](#) [Ipopt::OrigIterationOutput::~~OrigIterationOutput](#) () [[virtual](#)]

Default destructor.

6.124.2.3 [Ipopt::OrigIterationOutput::OrigIterationOutput](#) (const [OrigIterationOutput](#) &) [[private](#)]

Copy Constructor.

6.124.3 Member Function Documentation

6.124.3.1 [virtual](#) bool [Ipopt::OrigIterationOutput::Initializeml](#) (const [OptionsList](#) & *options*, const std::string & *prefix*) [[virtual](#)]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::IterationOutput](#).

6.124.3.2 `virtual void Ipopt::OrigIterationOutput::WriteOutput () [virtual]`

Method to do all the summary output per iteration.

This include the one-line summary output as well as writing the details about the iterates if desired

Implements [Ipopt::IterationOutput](#).

6.124.3.3 `static void Ipopt::OrigIterationOutput::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for [OptionsList](#).

6.124.3.4 `void Ipopt::OrigIterationOutput::operator= (const OrigIterationOutput &) [private]`

Overloaded Equals Operator.

6.124.4 Member Data Documentation

6.124.4.1 `bool Ipopt::OrigIterationOutput::print_info_string_ [private]`

Flag indicating weather info string should be printed at end of iteration summary line.

Definition at line 61 of file [IpOrigIterationOutput.hpp](#).

6.124.4.2 `InfPrOutput Ipopt::OrigIterationOutput::inf_pr_output_ [private]`

Option indication what should be printed in inf_pr column.

Definition at line 64 of file [IpOrigIterationOutput.hpp](#).

6.124.4.3 `int Ipopt::OrigIterationOutput::print_frequency_iter_ [private]`

Option indicating at which iteration frequency the summary line should be printed.

Definition at line 67 of file [IpOrigIterationOutput.hpp](#).

6.124.4.4 `Number Ipopt::OrigIterationOutput::print_frequency_time_ [private]`

Option indicating at which time frequency the summary line should be printed.

Definition at line 69 of file [IpOrigIterationOutput.hpp](#).

The documentation for this class was generated from the following file:

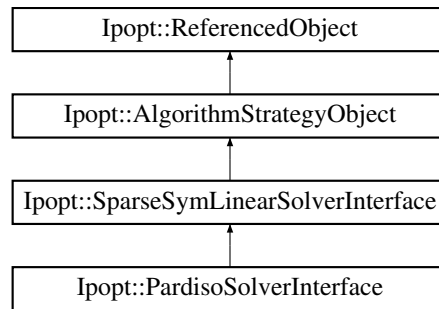
- [Algorithm/IpOrigIterationOutput.hpp](#)

6.125 Ipopt::PardisoSolverInterface Class Reference

Interface to the linear solver Pardiso, derived from [SparseSymLinearSolverInterface](#).

```
#include <IpPardisoSolverInterface.hpp>
```

Inheritance diagram for [Ipopt::PardisoSolverInterface](#):



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Constructor/Destructor

- [PardisoSolverInterface](#) ()
Constructor.
- virtual [~PardisoSolverInterface](#) ()
Destructor.

Methods for requesting solution of the linear system.

- virtual [ESymSolverStatus](#) [InitializeStructure](#) ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *ia, const [Index](#) *ja)
Method for initializing internal structures.
- virtual double * [GetValuesArrayPtr](#) ()
Method returning an internal array into which the nonzero elements are to be stored.
- virtual [ESymSolverStatus](#) [MultiSolve](#) (bool new_matrix, const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)
Solve operation for multiple right hand sides.
- virtual [Index](#) [NumberOfNegEVals](#) () const
Number of negative eigenvalues detected during last factorization.
- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.
- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat](#) [MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [PardisoSolverInterface](#) (const [PardisoSolverInterface](#) &)
Copy Constructor.
- void [operator=](#) (const [PardisoSolverInterface](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus SymbolicFactorization](#) (const [Index](#) *ia, const [Index](#) *ja)
Call Pardiso to do the analysis phase.
- [ESymSolverStatus Factorization](#) (const [Index](#) *ia, const [Index](#) *ja, bool check_NegEVals, [Index](#) numberOfNegEVals)
Call Pardiso to factorize the [Matrix](#).
- [ESymSolverStatus Solve](#) (const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals)
Call Pardiso to do the Solve.

Private Attributes

Information about the matrix

- [Index dim_](#)
Number of rows and columns of the matrix.
- [Index nonzeros_](#)
Number of nonzeros of the matrix in triplet representation.
- double * [a_](#)
Array for storing the values of the matrix.

Information about most recent factorization/solve

- [Index negevals_](#)
Number of negative eigenvalues.

Initialization flags

- bool [initialized_](#)
Flag indicating if internal data is initialized.

Solver specific information

- void ** [PT_](#)
Internal data address pointers.
- [ipfint MAXFCT_](#)
Maximal number of factors with identical nonzero structure.
- [ipfint MNUM_](#)
Actual matrix for the solution phase.
- [ipfint MTYPE_](#)
[Matrix](#) type; real and symmetric indefinite.

- `ipfint * IPARM_`
Parameter and info array for Pardiso.
- `double * DPARM_`
Parameter and info array for Pardiso.
- `ipfint MSGVLVL_`
Message level.

Some counters for debugging

- `Index debug_last_iter_`
- `Index debug_cnt_`

Solver specific options

- `enum PardisoMatchingStrategy { COMPLETE, COMPLETE2x2, CONSTRAINT }`
Type for mathcing strategies.
- `PardisoMatchingStrategy match_strat_`
Option that controls the matching strategy.
- `bool have_symbolic_factorization_`
Flag indicating if symbolic factorization has already been performed.
- `bool pardiso_redo_symbolic_fact_only_if_inertia_wrong_`
Flag indicating whether the symbolic factorization should only be done after perturbed elements, if the inertia was wrong.
- `bool pardiso_repeated_perturbation_means_singular_`
Flag indicating whether repeated perturbed elements even after a new symbolic factorization should be interpreted as a singular matrix.
- `bool skip_inertia_check_`
Flag indicating if the interia is always assumed to be correct.
- `bool pardiso_iterative_`
Flag indicating whether we are using the iterative solver in Pardiso.
- `Index pardiso_max_droptol_corrections_`
Maximal number of decreases of drop tolerance during one solve.

Additional Inherited Members

6.125.1 Detailed Description

Interface to the linear solver Pardiso, derived from [SparseSymLinearSolverInterface](#).

For details, see description of [SparseSymLinearSolverInterface](#) base class.

Definition at line 24 of file `IpPardisoSolverInterface.hpp`.

6.125.2 Member Enumeration Documentation

6.125.2.1 `enum Ipopt::PardisoSolverInterface::PardisoMatchingStrategy` [private]

Type for mathcing strategies.

Enumerator

COMPLETE

COMPLETE2x2**CONSTRAINT**

Definition at line 141 of file IpPardisoSolverInterface.hpp.

6.125.3 Constructor & Destructor Documentation

6.125.3.1 Ipopt::PardisoSolverInterface::PardisoSolverInterface ()

Constructor.

6.125.3.2 virtual Ipopt::PardisoSolverInterface::~~PardisoSolverInterface () [virtual]

Destructor.

6.125.3.3 Ipopt::PardisoSolverInterface::PardisoSolverInterface (const PardisoSolverInterface &) [private]

Copy Constructor.

6.125.4 Member Function Documentation

6.125.4.1 bool Ipopt::PardisoSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.125.4.2 virtual ESymSolverStatus Ipopt::PardisoSolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * ia, const Index * ja) [virtual]

Method for initializing internal stuctures.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.125.4.3 virtual double* Ipopt::PardisoSolverInterface::GetValuesArrayPtr () [virtual]

Method returing an internal array into which the nonzero elements are to be stored.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.125.4.4 virtual ESymSolverStatus Ipopt::PardisoSolverInterface::MultiSolve (bool new_matrix, const Index * ia, const Index * ja, Index nrhs, double * rhs_vals, bool check_NegEVals, Index numberOfNegEVals) [virtual]

Solve operation for multiple right hand sides.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.125.4.5 virtual Index Ipopt::PardisoSolverInterface::NumberOfNegEVals () const [virtual]

Number of negative eigenvalues detected during last factorization.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.125.4.6 virtual bool Ipopt::PardisoSolverInterface::IncreaseQuality () [virtual]

Request to increase quality of solution for next solve.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.125.4.7 `virtual bool Ipopt::PardisoSolverInterface::ProvidesInertia () const [inline],[virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 76 of file `IpPardisoSolverInterface.hpp`.

6.125.4.8 `EMatrixFormat Ipopt::PardisoSolverInterface::MatrixFormat () const [inline],[virtual]`

Query of requested matrix type that the linear solver understands.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 83 of file `IpPardisoSolverInterface.hpp`.

6.125.4.9 `static void Ipopt::PardisoSolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for `IpoptType`.

6.125.4.10 `void Ipopt::PardisoSolverInterface::operator= (const PardisoSolverInterface &) [private]`

Overloaded Equals Operator.

6.125.4.11 `ESymSolverStatus Ipopt::PardisoSolverInterface::SymbolicFactorization (const Index * ia, const Index * ja) [private]`

Call Pardiso to do the analysis phase.

6.125.4.12 `ESymSolverStatus Ipopt::PardisoSolverInterface::Factorization (const Index * ia, const Index * ja, bool check_NegEVals, Index numberOfNegEVals) [private]`

Call Pardiso to factorize the [Matrix](#).

6.125.4.13 `ESymSolverStatus Ipopt::PardisoSolverInterface::Solve (const Index * ia, const Index * ja, Index nrhs, double * rhs_vals) [private]`

Call Pardiso to do the Solve.

6.125.5 Member Data Documentation

6.125.5.1 `Index Ipopt::PardisoSolverInterface::dim_ [private]`

Number of rows and columns of the matrix.

Definition at line 113 of file `IpPardisoSolverInterface.hpp`.

6.125.5.2 `Index Ipopt::PardisoSolverInterface::nonzeros_ [private]`

Number of nonzeros of the matrix in triplet representation.

Definition at line 116 of file `IpPardisoSolverInterface.hpp`.

6.125.5.3 `double* Ipopt::PardisoSolverInterface::a_ [private]`

Array for storing the values of the matrix.

Definition at line 119 of file IpPardisoSolverInterface.hpp.

6.125.5.4 `Index Ipopt::PardisoSolverInterface::negevals_ [private]`

Number of negative eigenvalues.

Definition at line 135 of file IpPardisoSolverInterface.hpp.

6.125.5.5 `PardisoMatchingStrategy Ipopt::PardisoSolverInterface::match_strat_ [private]`

Option that controls the matching strategy.

Definition at line 148 of file IpPardisoSolverInterface.hpp.

6.125.5.6 `bool Ipopt::PardisoSolverInterface::have_symbolic_factorization_ [private]`

Flag indicating if symbolic factorization has already been performed.

Definition at line 151 of file IpPardisoSolverInterface.hpp.

6.125.5.7 `bool Ipopt::PardisoSolverInterface::pardiso_redo_symbolic_fact_only_if_inertia_wrong_ [private]`

Flag indicating whether the symbolic factorization should only be done after perturbed elements, if the inertia was wrong.

Definition at line 154 of file IpPardisoSolverInterface.hpp.

6.125.5.8 `bool Ipopt::PardisoSolverInterface::pardiso_repeated_perturbation_means_singular_ [private]`

Flag indicating whether repeated perturbed elements even after a new symbolic factorization should be interpreted as a singular matrix.

Definition at line 158 of file IpPardisoSolverInterface.hpp.

6.125.5.9 `bool Ipopt::PardisoSolverInterface::skip_inertia_check_ [private]`

Flag indicating if the inertia is always assumed to be correct.

Definition at line 161 of file IpPardisoSolverInterface.hpp.

6.125.5.10 `bool Ipopt::PardisoSolverInterface::pardiso_iterative_ [private]`

Flag indicating whether we are using the iterative solver in Pardiso.

Definition at line 164 of file IpPardisoSolverInterface.hpp.

6.125.5.11 `Index Ipopt::PardisoSolverInterface::pardiso_max_droptol_corrections_ [private]`

Maximal number of decreases of drop tolerance during one solve.

Definition at line 166 of file IpPardisoSolverInterface.hpp.

6.125.5.12 `bool Ipopt::PardisoSolverInterface::initialized_ [private]`

Flag indicating if internal data is initialized.

For initialization, this object needs to have seen a matrix

Definition at line 173 of file IpPardisoSolverInterface.hpp.

6.125.5.13 `void** Ipopt::PardisoSolverInterface::PT_ [private]`

Internal data address pointers.

Definition at line 179 of file `IpPardisoSolverInterface.hpp`.

6.125.5.14 `ipfint Ipopt::PardisoSolverInterface::MAXFCT_ [private]`

Maximal number of factors with identical nonzero structure.

Here, we only store one factorization. Is always 1.

Definition at line 182 of file `IpPardisoSolverInterface.hpp`.

6.125.5.15 `ipfint Ipopt::PardisoSolverInterface::MNUM_ [private]`

Actual matrix for the solution phase.

Is always 1.

Definition at line 184 of file `IpPardisoSolverInterface.hpp`.

6.125.5.16 `ipfint Ipopt::PardisoSolverInterface::MTYPE_ [private]`

[Matrix](#) type; real and symmetric indefinite.

Is always -2.

Definition at line 186 of file `IpPardisoSolverInterface.hpp`.

6.125.5.17 `ipfint* Ipopt::PardisoSolverInterface::IPARM_ [private]`

Parameter and info array for Pardiso.

Definition at line 188 of file `IpPardisoSolverInterface.hpp`.

6.125.5.18 `double* Ipopt::PardisoSolverInterface::DPARAM_ [private]`

Parameter and info array for Pardiso.

Definition at line 190 of file `IpPardisoSolverInterface.hpp`.

6.125.5.19 `ipfint Ipopt::PardisoSolverInterface::MSGVLV_ [private]`

Message level.

Definition at line 192 of file `IpPardisoSolverInterface.hpp`.

6.125.5.20 `Index Ipopt::PardisoSolverInterface::debug_last_iter_ [private]`

Definition at line 197 of file `IpPardisoSolverInterface.hpp`.

6.125.5.21 `Index Ipopt::PardisoSolverInterface::debug_cnt_ [private]`

Definition at line 198 of file `IpPardisoSolverInterface.hpp`.

The documentation for this class was generated from the following file:

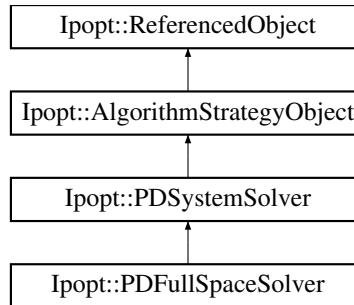
- [Algorithm/LinearSolvers/IpPardisoSolverInterface.hpp](#)

6.126 Ipopt::PDFullSpaceSolver Class Reference

This is the implemetation of the Primal-Dual System, using the full space approach with a direct linear solver.

```
#include <IpPDFullSpaceSolver.hpp>
```

Inheritance diagram for Ipopt::PDFullSpaceSolver:



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [Solve](#) ([Number](#) alpha, [Number](#) beta, const [IteratesVector](#) &rhs, [IteratesVector](#) &res, bool allow_inexact=false, bool improve_solution=false)
Solve the primal dual system, given one right hand side.

/Destructor

- [PDFullSpaceSolver](#) ([AugSystemSolver](#) &augSysSolver, [PDPerturbationHandler](#) &perturbHandler)
Constructor that takes in the Augmented System solver that is to be used inside.
- virtual [~PDFullSpaceSolver](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

- bool [SolveOnce](#) (bool resolve_unmodified, bool pretend_singular, const [SymMatrix](#) &W, const [Matrix](#) &J_c, const [Matrix](#) &J_d, const [Matrix](#) &Px_L, const [Matrix](#) &Px_U, const [Matrix](#) &Pd_L, const [Matrix](#) &Pd_U, const [Vector](#) &z_L, const [Vector](#) &z_U, const [Vector](#) &v_L, const [Vector](#) &v_U, const [Vector](#) &slack_x_L, const [Vector](#) &slack_x_U, const [Vector](#) &slack_s_L, const [Vector](#) &slack_s_U, const [Vector](#) &sigma_x, const [Vector](#) &sigma_s, [Number](#) alpha, [Number](#) beta, const [IteratesVector](#) &rhs, [IteratesVector](#) &res)
Internal function for a single backsolve (which will be used for iterative refinement on the outside).
- void [ComputeResiduals](#) (const [SymMatrix](#) &W, const [Matrix](#) &J_c, const [Matrix](#) &J_d, const [Matrix](#) &Px_L, const [Matrix](#) &Px_U, const [Matrix](#) &Pd_L, const [Matrix](#) &Pd_U, const [Vector](#) &z_L, const [Vector](#) &z_U, const [Vector](#) &v_L, const [Vector](#) &v_U, const [Vector](#) &slack_x_L, const [Vector](#) &slack_x_U, const [Vector](#) &slack_s_L, const [Vector](#) &slack_s_U, const [Vector](#) &sigma_x, const [Vector](#) &sigma_s, [Number](#) alpha, [Number](#) beta, const [IteratesVector](#) &rhs, const [IteratesVector](#) &res, [IteratesVector](#) &resid)

Internal function for computing the residual (resid) given the right hand side (rhs) and the solution of the system (res).

- **Number ComputeResidualRatio** (const **IteratesVector** &rhs, const **IteratesVector** &res, const **IteratesVector** &resid)

Internal function for computing the ratio of the residual compared to the right hand side and solution.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- **PDFullSpaceSolver** ()
Default Constructor.
- **PDFullSpaceSolver** & **operator=** (const **PDFullSpaceSolver** &)
Overloaded Equals Operator.

Auxilliary functions

- void **SinvBlrmZPTdBr** (**Number** alpha, const **Vector** &S, const **Vector** &R, const **Vector** &Z, const **Matrix** &P, const **Vector** &g, **Vector** &X)
Compute $x = S^{-1}(r + \alpha ZP^T d)$.

Private Attributes

Strategy objects to hold on to.

- **SmartPtr**< **AugSystemSolver** > **augSysSolver_**
Pointer to the Solver for the augmented system.
- **SmartPtr**< **PDPerturbationHandler** > **perturbHandler_**
Pointer to the Perturbation Handler.

Data about the correction made to the system

- **CachedResults**< void * > **dummy_cache_**
A dummy cache to figure out if the deltas are still up to date.
- bool **augsys_improved_**
Flag indicating if for the current matrix the solution quality of the augmented system solver has already been increased.

Parameters

- **Index min_refinement_steps_**
Minimal number of iterative refinement performed per backsolve.
- **Index max_refinement_steps_**
Maximal number of iterative refinement performed per backsolve.
- **Number residual_ratio_max_**
Maximal allowed ratio of the norm of the residual over the norm of the right hand side and solution.
- **Number residual_ratio_singular_**
If the residual_ratio is larger than this value after trying to improve the solution, the linear system is assumed to be singular and modified.
- **Number residual_improvement_factor_**
Factor defining require improvement to consider iterative refinement successful.
- **Number neg_curv_test_tol_**
Tolernace for heuristic to ignore wrong inertia.

Additional Inherited Members

6.126.1 Detailed Description

This is the implemetation of the Primal-Dual System, using the full space approach with a direct linear solver.

A note on the iterative refinement: We perform at least `min_refinement_steps` number of iterative refinement steps. If after one iterative refinement the quality of the solution (defined in `ResidualRatio`) does not improve or the maximal number of iterative refinement steps is exceeded before the tolerance `residual_ratio_max_` is satisfied, we first ask the linear solver to solve the system more accurately (e.g. by increasing the pivot tolerance). If that doesn't help or is not possible, we treat the system, as if it is singular (i.e. increase delta's).

Definition at line 32 of file `IpPDFullSpaceSolver.hpp`.

6.126.2 Constructor & Destructor Documentation

6.126.2.1 Ipopt::PDFullSpaceSolver::PDFullSpaceSolver (AugSystemSolver & *augSysSolver*, PDPerturbationHandler & *perturbHandler*)

Constructor that takes in the Augmented System solver that is to be used inside.

6.126.2.2 virtual Ipopt::PDFullSpaceSolver::~~PDFullSpaceSolver () [virtual]

Default destructor.

6.126.2.3 Ipopt::PDFullSpaceSolver::PDFullSpaceSolver () [private]

Default Constructor.

6.126.3 Member Function Documentation

6.126.3.1 bool Ipopt::PDFullSpaceSolver::InitializeImpl (const OptionsList & *options*, const std::string & *prefix*) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::PDSolver](#).

6.126.3.2 virtual bool Ipopt::PDFullSpaceSolver::Solve (Number *alpha*, Number *beta*, const IteratesVector & *rhs*, IteratesVector & *res*, bool *allow_inexact* = false, bool *improve_solution* = false) [virtual]

Solve the primal dual system, given one right hand side.

Implements [Ipopt::PDSolver](#).

6.126.3.3 static void Ipopt::PDFullSpaceSolver::RegisterOptions (SmartPtr< RegisteredOptions > *options*) [static]

Methods for `IpoptType`.

6.126.3.4 PDFullSpaceSolver& Ipopt::PDFullSpaceSolver::operator= (const PDFullSpaceSolver &) [private]

Overloaded Equals Operator.

6.126.3.5 **bool** Ipopt::PDFullSpaceSolver::SolveOnce (**bool** *resolve_unmodified*, **bool** *pretend_singular*, **const** **SymMatrix** & *W*, **const** **Matrix** & *J_c*, **const** **Matrix** & *J_d*, **const** **Matrix** & *Px_L*, **const** **Matrix** & *Px_U*, **const** **Matrix** & *Pd_L*, **const** **Matrix** & *Pd_U*, **const** **Vector** & *z_L*, **const** **Vector** & *z_U*, **const** **Vector** & *v_L*, **const** **Vector** & *v_U*, **const** **Vector** & *slack_x_L*, **const** **Vector** & *slack_x_U*, **const** **Vector** & *slack_s_L*, **const** **Vector** & *slack_s_U*, **const** **Vector** & *sigma_x*, **const** **Vector** & *sigma_s*, **Number** *alpha*, **Number** *beta*, **const** **IteratesVector** & *rhs*, **IteratesVector** & *res*)
[private]

Internal function for a single backsolve (which will be used for iterative refinement on the outside).

This method returns false, if for some reason the linear system could not be solved (e.g. when the regularization parameter becomes too large.)

6.126.3.6 **void** Ipopt::PDFullSpaceSolver::ComputeResiduals (**const** **SymMatrix** & *W*, **const** **Matrix** & *J_c*, **const** **Matrix** & *J_d*, **const** **Matrix** & *Px_L*, **const** **Matrix** & *Px_U*, **const** **Matrix** & *Pd_L*, **const** **Matrix** & *Pd_U*, **const** **Vector** & *z_L*, **const** **Vector** & *z_U*, **const** **Vector** & *v_L*, **const** **Vector** & *v_U*, **const** **Vector** & *slack_x_L*, **const** **Vector** & *slack_x_U*, **const** **Vector** & *slack_s_L*, **const** **Vector** & *slack_s_U*, **const** **Vector** & *sigma_x*, **const** **Vector** & *sigma_s*, **Number** *alpha*, **Number** *beta*, **const** **IteratesVector** & *rhs*, **const** **IteratesVector** & *res*, **IteratesVector** & *resid*) [private]

Internal function for computing the residual (resid) given the right hand side (rhs) and the solution of the system (res).

6.126.3.7 **Number** Ipopt::PDFullSpaceSolver::ComputeResidualRatio (**const** **IteratesVector** & *rhs*, **const** **IteratesVector** & *res*, **const** **IteratesVector** & *resid*) [private]

Internal function for computing the ratio of the residual compared to the right hand side and solution.

The smaller this value, the better the solution.

6.126.3.8 **void** Ipopt::PDFullSpaceSolver::SinvBlrmZPTdBr (**Number** *alpha*, **const** **Vector** & *S*, **const** **Vector** & *R*, **const** **Vector** & *Z*, **const** **Matrix** & *P*, **const** **Vector** & *g*, **Vector** & *X*) [private]

Compute $x = S^{-1}(r + \alpha ZP^T d)$.

6.126.4 Member Data Documentation

6.126.4.1 **SmartPtr**<**AugSystemSolver**> Ipopt::PDFullSpaceSolver::augSysSolver_ [private]

Pointer to the Solver for the augmented system.

Definition at line 83 of file IpPDFullSpaceSolver.hpp.

6.126.4.2 **SmartPtr**<**PDPerturbationHandler**> Ipopt::PDFullSpaceSolver::perturbHandler_ [private]

Pointer to the Perturbation Handler.

Definition at line 85 of file IpPDFullSpaceSolver.hpp.

6.126.4.3 **CachedResults**<**void***> Ipopt::PDFullSpaceSolver::dummy_cache_ [private]

A dummy cache to figure out if the deltas are still up to date.

Definition at line 91 of file IpPDFullSpaceSolver.hpp.

6.126.4.4 **bool** Ipopt::PDFullSpaceSolver::augsys_improved_ [private]

Flag indicating if for the current matrix the solution quality of the augmented system solver has already been increased.

Definition at line 94 of file IpPDFullSpaceSolver.hpp.

6.126.4.5 Index Ipopt::PDFullSpaceSolver::min_refinement_steps_ [private]

Minimal number of iterative refinement performed per backsolve.

Definition at line 100 of file IpPDFullSpaceSolver.hpp.

6.126.4.6 Index Ipopt::PDFullSpaceSolver::max_refinement_steps_ [private]

Maximal number of iterative refinement performed per backsolve.

Definition at line 102 of file IpPDFullSpaceSolver.hpp.

6.126.4.7 Number Ipopt::PDFullSpaceSolver::residual_ratio_max_ [private]

Maximal allowed ratio of the norm of the residual over the norm of the right hand side and solution.

Definition at line 105 of file IpPDFullSpaceSolver.hpp.

6.126.4.8 Number Ipopt::PDFullSpaceSolver::residual_ratio_singular_ [private]

If the residual_ratio is larger than this value after trying to improve the solution, the linear system is assumed to be singular and modified.

Definition at line 109 of file IpPDFullSpaceSolver.hpp.

6.126.4.9 Number Ipopt::PDFullSpaceSolver::residual_improvement_factor_ [private]

Factor defining require improvement to consider iterative refinement successful.

Definition at line 112 of file IpPDFullSpaceSolver.hpp.

6.126.4.10 Number Ipopt::PDFullSpaceSolver::neg_curv_test_tol_ [private]

Tolernace for heuristic to ignore wrong inertia.

Definition at line 114 of file IpPDFullSpaceSolver.hpp.

The documentation for this class was generated from the following file:

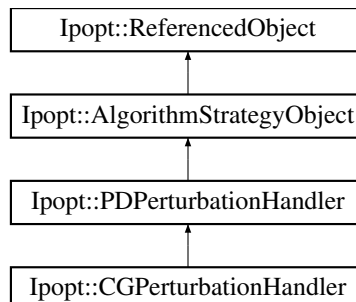
- [Algorithm/IpPDFullSpaceSolver.hpp](#)

6.127 Ipopt::PDPerturbationHandler Class Reference

Class for handling the perturbation factors delta_x, delta_s, delta_c, and delta_d in the primal dual system.

```
#include <IpPDPerturbationHandler.hpp>
```

Inheritance diagram for Ipopt::PDPerturbationHandler:



Public Member Functions

- virtual bool `InitializeImpl` (const `OptionsList` &options, const std::string &prefix)
Implementation of the initialization method that has to be overloaded by for each derived class.
- virtual bool `ConsiderNewSystem` (Number &delta_x, Number &delta_s, Number &delta_c, Number &delta_d)
This method must be called for each new matrix, and before any other method for generating perturbation factors.
- virtual bool `PerturbForSingularity` (Number &delta_x, Number &delta_s, Number &delta_c, Number &delta_d)
This method returns perturbation factors for the case when the most recent factorization resulted in a singular matrix.
- virtual bool `PerturbForWrongInertia` (Number &delta_x, Number &delta_s, Number &delta_c, Number &delta_d)
This method returns perturbation factors for the case when the most recent factorization resulted in a matrix with an incorrect number of negative eigenvalues.
- virtual void `CurrentPerturbation` (Number &delta_x, Number &delta_s, Number &delta_c, Number &delta_d)
Just return the perturbation values that have been determined most recently.

Constructors/Destructors

- `PDPerturbationHandler` ()
Default Constructor.
- virtual `~PDPerturbationHandler` ()
Default destructor.

Static Public Member Functions

- static void `RegisterOptions` (SmartPtr< `RegisteredOptions` > roptions)
Methods for lpoptType.

Protected Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `PDPerturbationHandler` (const `PDPerturbationHandler` &)
Copy Constructor.
- void `operator=` (const `PDPerturbationHandler` &)
Overloaded Equals Operator.

Auxilliary methods

- bool `get_deltas_for_wrong_inertia` (Number &delta_x, Number &delta_s, Number &delta_c, Number &delta_d)
Internal version of PerturbForWrongInertia with the difference, that finalize_test is not called.
- void `finalize_test` ()
This method is call whenever a matrix had been factorization and is not singular.
- Number `delta_cd` ()
Compute perturbation value for constraints.

Protected Attributes

- bool [get_deltas_for_wrong_inertia_called_](#)
Flag indicating if for the given matrix the perturb for wrong inertia method has already been called.

Size of the most recent non-zero perturbation.

- Number [delta_x_last_](#)
The last nonzero value for delta_x.
- Number [delta_s_last_](#)
The last nonzero value for delta_s.
- Number [delta_c_last_](#)
The last nonzero value for delta_c.
- Number [delta_d_last_](#)
The last nonzero value for delta_d.

Size of the most recently suggested perturbation for the current matrix.

- Number [delta_x_curr_](#)
The current value for delta_x.
- Number [delta_s_curr_](#)
The current value for delta_s.
- Number [delta_c_curr_](#)
The current value for delta_c.
- Number [delta_d_curr_](#)
The current value for delta_d.

Algorithmic parameters.

- Number [delta_xs_max_](#)
Maximal perturbation for x and s.
- Number [delta_xs_min_](#)
Smallest possible perturbation for x and s.
- Number [delta_xs_first_inc_fact_](#)
Increase factor for delta_xs for first required perturbation.
- Number [delta_xs_inc_fact_](#)
Increase factor for delta_xs for later perturbations.
- Number [delta_xs_dec_fact_](#)
Decrease factor for delta_xs for later perturbations.
- Number [delta_xs_init_](#)
Very first trial value for delta_xs perturbation.
- Number [delta_cd_val_](#)
Size of perturbation for c and d blocks.
- Number [delta_cd_exp_](#)
Exponent on mu in formula for of perturbation for c and d blocks.
- bool [reset_last_](#)
Flag indicating whether the new values are based on the perturbations in the last iteration or in the more recent iteration in which a perturbation was done.
- Index [degen_iters_max_](#)
Required number of iterations for degeneracy conclusions.
- bool [perturb_always_cd_](#)
Flag indicating that the delta_c, delta_d perturbation should always be used.

Handling structural degeneracy

- enum `DegenType` { `NOT_YET_DETERMINED`, `NOT_DEGENERATE`, `DEGENERATE` }
Type for degeneracy flags.
- enum `TrialStatus` {
 `NO_TEST`, `TEST_DELTA_C_EQ_0_DELTA_X_EQ_0`, `TEST_DELTA_C_GT_0_DELTA_X_EQ_0`, `TEST_DELTA_C_EQ_0_DELTA_X_GT_0`,
 `TEST_DELTA_C_GT_0_DELTA_X_GT_0` }
Status of current trial configuration.
- `DegenType hess_degenerate_`
Flag indicating whether the reduced Hessian matrix is thought to be structurally singular.
- `DegenType jac_degenerate_`
Flag indicating whether the Jacobian of the constraints is thought to be structurally rank-deficient.
- `Index degen_iters_`
Flag counting matrices in which degeneracy was observed in the first successive iterations.
- `TrialStatus test_status_`
Current status.

6.127.1 Detailed Description

Class for handling the perturbation factors `delta_x`, `delta_s`, `delta_c`, and `delta_d` in the primal dual system.

This class is used by the `PDFullSpaceSolver` to handle the cases where the primal-dual system is singular or has the wrong inertia. The perturbation factors are obtained based on simple heuristics, taking into account the size of previous perturbations.

Definition at line 24 of file `IpPDPerturbationHandler.hpp`.

6.127.2 Member Enumeration Documentation

6.127.2.1 enum `Ipopt::PDPerturbationHandler::DegenType` [protected]

Type for degeneracy flags.

Enumerator

`NOT_YET_DETERMINED`
`NOT_DEGENERATE`
`DEGENERATE`

Definition at line 121 of file `IpPDPerturbationHandler.hpp`.

6.127.2.2 enum `Ipopt::PDPerturbationHandler::TrialStatus` [protected]

Status of current trial configuration.

Enumerator

`NO_TEST`
`TEST_DELTA_C_EQ_0_DELTA_X_EQ_0`
`TEST_DELTA_C_GT_0_DELTA_X_EQ_0`

TEST_DELTA_C_EQ_0_DELTA_X_GT_0

TEST_DELTA_C_GT_0_DELTA_X_GT_0

Definition at line 142 of file IpPDPerturbationHandler.hpp.

6.127.3 Constructor & Destructor Documentation

6.127.3.1 Ipopt::PDPerturbationHandler::PDPerturbationHandler ()

Default Constructor.

6.127.3.2 virtual Ipopt::PDPerturbationHandler::~~PDPerturbationHandler () [inline], [virtual]

Default destructor.

Definition at line 32 of file IpPDPerturbationHandler.hpp.

6.127.3.3 Ipopt::PDPerturbationHandler::PDPerturbationHandler (const PDPerturbationHandler &) [protected]

Copy Constructor.

6.127.4 Member Function Documentation

6.127.4.1 virtual bool Ipopt::PDPerturbationHandler::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

Implementation of the initialization method that has to be overloaded by for each derived class.

Implements [Ipopt::AlgorithmStrategyObject](#).

Reimplemented in [Ipopt::CGPerturbationHandler](#).

6.127.4.2 virtual bool Ipopt::PDPerturbationHandler::ConsiderNewSystem (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [virtual]

This method must be called for each new matrix, and before any other method for generating perturbation factors.

Usually, the returned perturbation factors are zero, but if the system is thought to be structurally singular, they might be positive. If the return value is false, no suitable perturbation could be found.

Reimplemented in [Ipopt::CGPerturbationHandler](#).

6.127.4.3 virtual bool Ipopt::PDPerturbationHandler::PerturbForSingularity (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [virtual]

This method returns perturbation factors for the case when the most recent factorization resulted in a singular matrix.

If the return value is false, no suitable perturbation could be found.

Reimplemented in [Ipopt::CGPerturbationHandler](#).

6.127.4.4 virtual bool Ipopt::PDPerturbationHandler::PerturbForWrongInertia (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [virtual]

This method returns perturbation factors for the case when the most recent factorization resulted in a matrix with an incorrect number of negative eigenvalues.

If the return value is false, no suitable perturbation could be found.

Reimplemented in [Ipopt::CGPerturbationHandler](#).

6.127.4.5 `virtual void Ipopt::PDPerturbationHandler::CurrentPerturbation (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [virtual]`

Just return the perturbation values that have been determined most recently.

Reimplemented in [Ipopt::CGPerturbationHandler](#).

6.127.4.6 `static void Ipopt::PDPerturbationHandler::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for IpoptType.

6.127.4.7 `void Ipopt::PDPerturbationHandler::operator=(const PDPerturbationHandler &) [protected]`

Overloaded Equals Operator.

6.127.4.8 `bool Ipopt::PDPerturbationHandler::get_deltas_for_wrong_inertia (Number & delta_x, Number & delta_s, Number & delta_c, Number & delta_d) [protected]`

Internal version of PerturbForWrongInertia with the difference, that finalize_test is not called.

Returns false if the delta_x and delta_s parameters become too large.

6.127.4.9 `void Ipopt::PDPerturbationHandler::finalize_test () [protected]`

This method is call whenever a matrix had been factorization and is not singular.

In here, we can evaluate the outcome of the denegeracy test heuristics.

6.127.4.10 `Number Ipopt::PDPerturbationHandler::delta_cd () [protected]`

Compute perturbation value for constraints.

6.127.5 Member Data Documentation

6.127.5.1 `Number Ipopt::PDPerturbationHandler::delta_x_last_ [protected]`

The last nonzero value for delta_x.

Definition at line 92 of file IpPDPerturbationHandler.hpp.

6.127.5.2 `Number Ipopt::PDPerturbationHandler::delta_s_last_ [protected]`

The last nonzero value for delta_s.

Definition at line 94 of file IpPDPerturbationHandler.hpp.

6.127.5.3 `Number Ipopt::PDPerturbationHandler::delta_c_last_ [protected]`

The last nonzero value for delta_c.

Definition at line 96 of file IpPDPerturbationHandler.hpp.

6.127.5.4 `Number Ipopt::PDPerturbationHandler::delta_d_last_ [protected]`

The last nonzero value for delta_d.

Definition at line 98 of file IpPDPerturbationHandler.hpp.

6.127.5.5 Number Ipopt::PDPerturbationHandler::delta_x_curr_ [protected]

The current value for delta_x.

Definition at line 105 of file IpPDPerturbationHandler.hpp.

6.127.5.6 Number Ipopt::PDPerturbationHandler::delta_s_curr_ [protected]

The current value for delta_s.

Definition at line 107 of file IpPDPerturbationHandler.hpp.

6.127.5.7 Number Ipopt::PDPerturbationHandler::delta_c_curr_ [protected]

The current value for delta_c.

Definition at line 109 of file IpPDPerturbationHandler.hpp.

6.127.5.8 Number Ipopt::PDPerturbationHandler::delta_d_curr_ [protected]

The current value for delta_d.

Definition at line 111 of file IpPDPerturbationHandler.hpp.

6.127.5.9 bool Ipopt::PDPerturbationHandler::get_deltas_for_wrong_inertia_called_ [protected]

Flag indicating if for the given matrix the perturb for wrong inertia method has already been called.

Definition at line 116 of file IpPDPerturbationHandler.hpp.

6.127.5.10 DegenType Ipopt::PDPerturbationHandler::hess_degenerate_ [protected]

Flag indicating whether the reduced Hessian matrix is thought to be structurally singular.

Definition at line 130 of file IpPDPerturbationHandler.hpp.

6.127.5.11 DegenType Ipopt::PDPerturbationHandler::jac_degenerate_ [protected]

Flag indicating whether the Jacobian of the constraints is thought to be structurally rank-deficient.

Definition at line 134 of file IpPDPerturbationHandler.hpp.

6.127.5.12 Index Ipopt::PDPerturbationHandler::degen_iters_ [protected]

Flag counting matrices in which degeneracy was observed in the first successive iterations.

-1 means that there was a non-degenerate (unperturbed) matrix at some point.

Definition at line 139 of file IpPDPerturbationHandler.hpp.

6.127.5.13 TrialStatus Ipopt::PDPerturbationHandler::test_status_ [protected]

Current status.

Definition at line 152 of file IpPDPerturbationHandler.hpp.

6.127.5.14 Number Ipopt::PDPerturbationHandler::delta_xs_max_ [protected]

Maximal perturbation for x and s.

Definition at line 158 of file IpPDPerturbationHandler.hpp.

6.127.5.15 **Number** `lpopt::PDPerturbationHandler::delta_xs_min_` `[protected]`

Smallest possible perturbation for x and s.

Definition at line 160 of file `lpPDPerturbationHandler.hpp`.

6.127.5.16 **Number** `lpopt::PDPerturbationHandler::delta_xs_first_inc_fact_` `[protected]`

Increase factor for `delta_xs` for first required perturbation.

Definition at line 162 of file `lpPDPerturbationHandler.hpp`.

6.127.5.17 **Number** `lpopt::PDPerturbationHandler::delta_xs_inc_fact_` `[protected]`

Increase factor for `delta_xs` for later perturbations.

Definition at line 164 of file `lpPDPerturbationHandler.hpp`.

6.127.5.18 **Number** `lpopt::PDPerturbationHandler::delta_xs_dec_fact_` `[protected]`

Decrease factor for `delta_xs` for later perturbations.

Definition at line 166 of file `lpPDPerturbationHandler.hpp`.

6.127.5.19 **Number** `lpopt::PDPerturbationHandler::delta_xs_init_` `[protected]`

Very first trial value for `delta_xs` perturbation.

Definition at line 168 of file `lpPDPerturbationHandler.hpp`.

6.127.5.20 **Number** `lpopt::PDPerturbationHandler::delta_cd_val_` `[protected]`

Size of perturbation for c and d blocks.

Definition at line 170 of file `lpPDPerturbationHandler.hpp`.

6.127.5.21 **Number** `lpopt::PDPerturbationHandler::delta_cd_exp_` `[protected]`

Exponent on `mu` in formula for of perturbation for c and d blocks.

Definition at line 172 of file `lpPDPerturbationHandler.hpp`.

6.127.5.22 **bool** `lpopt::PDPerturbationHandler::reset_last_` `[protected]`

Flag indicating whether the new values are based on the perturbations in the last iteration or in the more recent iteration in which a perturbation was done.

Definition at line 176 of file `lpPDPerturbationHandler.hpp`.

6.127.5.23 **Index** `lpopt::PDPerturbationHandler::degen_iters_max_` `[protected]`

Required number of iterations for degeneracy conclusions.

Definition at line 178 of file `lpPDPerturbationHandler.hpp`.

6.127.5.24 **bool** `lpopt::PDPerturbationHandler::perturb_always_cd_` `[protected]`

Flag indicating that the `delta_c`, `delta_d` perturbation should always be used.

Definition at line 181 of file `lpPDPerturbationHandler.hpp`.

The documentation for this class was generated from the following file:

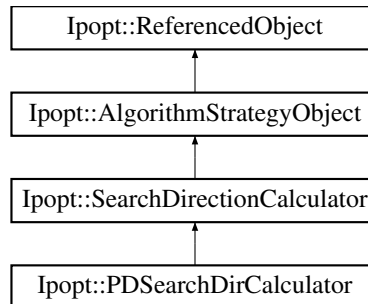
- [Algorithm/IpPDPerturbationHandler.hpp](#)

6.128 Ipopt::PDSearchDirCalculator Class Reference

Implementation of the search direction calculator that computes the pure primal dual step for the current barrier parameter.

```
#include <IpPDSearchDirCalc.hpp>
```

Inheritance diagram for Ipopt::PDSearchDirCalculator:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [ComputeSearchDirection](#) ()
Method for computing the search direction.
- [SmartPtr](#)< [PDSystemSolver](#) > [PDSolver](#) ()
Method to return the pd_solver for additional processing.

Constructors/Destructors

- [PDSearchDirCalculator](#) (const [SmartPtr](#)< [PDSystemSolver](#) > &pd_solver)
Constructor.
- virtual [~PDSearchDirCalculator](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (const [SmartPtr](#)< [RegisteredOptions](#) > &roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [PDSearchDirCalculator](#) ()
Default Constructor.
- [PDSearchDirCalculator](#) (const [PDSearchDirCalculator](#) &)
Copy Constructor.
- void [operator=](#) (const [PDSearchDirCalculator](#) &)
Overloaded Equals Operator.

Private Attributes

Strategy objects

- [SmartPtr](#)< [PDSystemSolver](#) > [pd_solver_](#)

Algorithmic parameters

- bool [fast_step_computation_](#)
Flag indicating that we trust that the steps from the linear solver are very good and that we don't need any residual checks.
- bool [mehrotra_algorithm_](#)
Flag indicating if we want to do Mehrotras's algorithm.

Additional Inherited Members

6.128.1 Detailed Description

Implementation of the search direction calculator that computes the pure primal dual step for the current barrier parameter.

Definition at line 21 of file `IpPDSearchDirCalc.hpp`.

6.128.2 Constructor & Destructor Documentation

6.128.2.1 `Ipopt::PDSearchDirCalculator::PDSearchDirCalculator (const SmartPtr< PDSystemSolver > & pd_solver)`

Constructor.

6.128.2.2 `virtual Ipopt::PDSearchDirCalculator::~~PDSearchDirCalculator () [virtual]`

Default destructor.

6.128.2.3 `Ipopt::PDSearchDirCalculator::PDSearchDirCalculator () [private]`

Default Constructor.

6.128.2.4 `Ipopt::PDSearchDirCalculator::PDSearchDirCalculator (const PDSearchDirCalculator &) [private]`

Copy Constructor.

6.128.3 Member Function Documentation

6.128.3.1 `virtual bool Ipopt::PDSearchDirCalculator::InitializImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SearchDirectionCalculator](#).

6.128.3.2 `virtual bool Ipopt::PDSearchDirCalculator::ComputeSearchDirection () [virtual]`

Method for computing the search direction.

The computed direction is stored in [IpData\(\).delta\(\)](#).

Implements [Ipopt::SearchDirectionCalculator](#).

6.128.3.3 `static void Ipopt::PDSearchDirCalculator::RegisterOptions (const SmartPtr< RegisteredOptions > & roptions) [static]`

Methods for IpoptType.

6.128.3.4 `SmartPtr<PDSystemSolver> Ipopt::PDSearchDirCalculator::PDSolver () [inline]`

Method to return the `pd_solver` for additional processing.

Definition at line 47 of file `IpPDSearchDirCalc.hpp`.

6.128.3.5 `void Ipopt::PDSearchDirCalculator::operator= (const PDSearchDirCalculator &) [private]`

Overloaded Equals Operator.

6.128.4 Member Data Documentation

6.128.4.1 `SmartPtr<PDSystemSolver> Ipopt::PDSearchDirCalculator::pd_solver_ [private]`

Definition at line 73 of file `IpPDSearchDirCalc.hpp`.

6.128.4.2 `bool Ipopt::PDSearchDirCalculator::fast_step_computation_ [private]`

Flag indicating that we trust that the steps from the linear solver are very good and that we don't need any residual checks.

Definition at line 81 of file `IpPDSearchDirCalc.hpp`.

6.128.4.3 `bool Ipopt::PDSearchDirCalculator::mehrotra_algorithm_ [private]`

Flag indicating if we want to do Mehrotra's algorithm.

This means that a number of options are ignored, or have to be set (or are automatically set) to certain values.

Definition at line 85 of file `IpPDSearchDirCalc.hpp`.

The documentation for this class was generated from the following file:

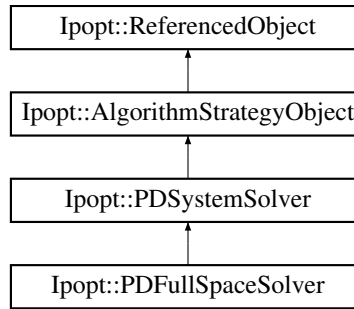
- [Algorithm/IpPDSearchDirCalc.hpp](#)

6.129 Ipopt::PDSystemSolver Class Reference

Pure Primal Dual System Solver Base Class.

```
#include <IpPDSystemSolver.hpp>
```

Inheritance diagram for `Ipopt::PDSystemSolver`:



Public Member Functions

- virtual bool **InitializeImpl** (const **OptionsList** &options, const std::string &prefix)=0
*overloaded from **AlgorithmStrategyObject***
- virtual bool **Solve** (**Number** alpha, **Number** beta, const **IteratesVector** &rhs, **IteratesVector** &res, bool allow_inexact=false, bool improve_solution=false)=0
Solve the primal dual system, given one right hand side.

/Destructor

- **PDSystemSolver** ()
Default Constructor.
- virtual **~PDSystemSolver** ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- **PDSystemSolver** & **operator=** (const **PDSystemSolver** &)
Overloaded Equals Operator.

Additional Inherited Members

6.129.1 Detailed Description

Pure Primal Dual System Solver Base Class.

This is the base class for all derived Primal-Dual System Solver Types.

Here, we understand the primal-dual system as the following linear system:

$$\begin{bmatrix}
 W & 0 & J_c^T & J_d^T & -P_L^x & P_U^x & 0 & 0 \\
 0 & 0 & 0 & -I & 0 & 0 & -P_L^d & P_U^d \\
 J_c & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 J_d & -I & 0 & 0 & 0 & 0 & 0 & 0 \\
 Z_L(P_L^x)^T & 0 & 0 & 0 & S_L^x & 0 & 0 & 0 \\
 -Z_U(P_U^x)^T & 0 & 0 & 0 & 0 & S_U^x & 0 & 0 \\
 0 & V_L(P_L^d)^T & 0 & 0 & 0 & 0 & S_L^s & 0 \\
 0 & -V_U(P_U^d)^T & 0 & 0 & 0 & 0 & 0 & S_U^s
 \end{bmatrix}
 \begin{pmatrix}
 sol_x \\
 sol_s \\
 sol_c \\
 sol_d \\
 sol_L^z \\
 sol_U^z \\
 sol_L^v \\
 sol_U^v
 \end{pmatrix}
 =
 \begin{pmatrix}
 rhs_x \\
 rhs_s \\
 rhs_c \\
 rhs_d \\
 rhs_L^z \\
 rhs_U^z \\
 rhs_L^v \\
 rhs_U^v
 \end{pmatrix}$$

Here, $Sl_L^x = (P_L^x)^T x - x_L$, $Sl_U^x = x_U - (P_U^x)^T x$, $Sl_L^d = (P_L^d)^T d(x) - d_L$, $Sl_U^d = d_U - (P_U^d)^T d(x)$. The results returned to the caller is $res = \alpha * sol + \beta * res$.

The solution of this linear system (in order to compute the search direction of the algorithm) usually requires a considerable amount of computation time. Therefore, it is important to tailor the solution of this system to the characteristics of the problem. The purpose of this base class is to provide a generic interface to the algorithm that it can use whenever it requires a solution of the above system. Particular implementation can then be written to provide the methods defined here.

It is implicitly assumed here, that the upper left 2 by 2 block is possibly modified (implicitly or explicitly) so that its projection onto the null space of the overall constraint Jacobian $\begin{bmatrix} J_c & 0 \\ J_d & -I \end{bmatrix}$ is positive definite. This is necessary to guarantee certain descent properties of the resulting search direction. For example, in the full space implementation, a multiple of the identity might be added to the upper left 2 by 2 block.

Note that the Solve method might be called several times for different right hand sides, but with identical data. Therefore, if possible, an implementation of PDSystem should check whether the incoming data has changed, and not redo factorization etc. unless necessary.

Definition at line 76 of file IpPDSystemSolver.hpp.

6.129.2 Constructor & Destructor Documentation

6.129.2.1 Ipopt::PDSystemSolver::PDSystemSolver () [inline]

Default Constructor.

Definition at line 82 of file IpPDSystemSolver.hpp.

6.129.2.2 virtual Ipopt::PDSystemSolver::~~PDSystemSolver () [inline],[virtual]

Default destructor.

Definition at line 86 of file IpPDSystemSolver.hpp.

6.129.3 Member Function Documentation

6.129.3.1 virtual bool Ipopt::PDSystemSolver::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::PDFullSpaceSolver](#).

6.129.3.2 virtual bool Ipopt::PDSystemSolver::Solve (Number alpha, Number beta, const IteratesVector & rhs, IteratesVector & res, bool allow_inexact = false, bool improve_solution = false) [pure virtual]

Solve the primal dual system, given one right hand side.

If the flag allow_inexact is set to true, it is not necessary to solve the system to best accuracy; for example, we don't want iterative refinement during the computation of the second order correction. On the other hand, if improve_solution is true, the solution given in res should be improved (here beta has to be zero, and res is assume to be the solution for the system using rhs, without the factor alpha...). The return value is false, if a solution could not be computed (for example, when the Hessian regularization parameter becomes too large.)

Implemented in [Ipopt::PDFullSpaceSolver](#).

6.129.3.3 PDSolverSolver& Ipopt::PDSolverSolver::operator= (const PDSolverSolver &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

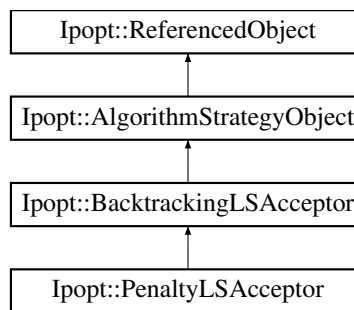
- [Algorithm/IpPDSolverSolver.hpp](#)

6.130 Ipopt::PenaltyLSAcceptor Class Reference

Penalty function line search.

```
#include <IpPenaltyLSAcceptor.hpp>
```

Inheritance diagram for Ipopt::PenaltyLSAcceptor:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
InitializeImpl - overloaded from [AlgorithmStrategyObject](#).
- virtual void [Reset](#) ()
Reset the acceptor.
- virtual void [InitThisLineSearch](#) (bool in_watchdog)
Initialization for the next line search.
- virtual void [PrepareRestoPhaseStart](#) ()
Method that is called before the restoration phase is called.
- virtual [Number](#) [CalculateAlphaMin](#) ()
Method returning the lower bound on the trial step sizes.
- virtual bool [CheckAcceptabilityOfTrialPoint](#) ([Number](#) alpha_primal)
Method for checking if current trial point is acceptable.
- virtual bool [TrySecondOrderCorrection](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try a second order correction for the constraints.
- virtual bool [TryCorrector](#) ([Number](#) alpha_primal_test, [Number](#) &alpha_primal, [SmartPtr](#)< [IteratesVector](#) > &actual_delta)
Try higher order corrector (for fast local convergence).
- virtual char [UpdateForNextIteration](#) ([Number](#) alpha_primal_test)
Method for ending the current line search.
- virtual void [StartWatchDog](#) ()
Method for setting internal data if the watchdog procedure is started.

- virtual void [StopWatchDog](#) ()
Method for setting internal data if the watchdog procedure is stopped.

Constructors/Destructors

- [PenaltyLSAcceptor](#) (const [SmartPtr](#)< [PDSystemSolver](#) > &pd_solver)
Constructor.
- virtual [~PenaltyLSAcceptor](#) ()
Default destructor.

Trial Point Accepting Methods. Used internally to check certain

acceptability criteria and used externally (by the restoration phase convergence check object, for instance)

- bool [IsAcceptableToCurrentIterate](#) ([Number](#) trial_barr, [Number](#) trial_theta, bool called_from_restoration=false)
const
Checks if a trial point is acceptable to the current iterate.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for [OptionsList](#).

Private Member Functions

- [Number](#) [CalcPred](#) ([Number](#) alpha)
Compute predicted reduction for given step size.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [PenaltyLSAcceptor](#) (const [PenaltyLSAcceptor](#) &)
Copy Constructor.
- void [operator=](#) (const [PenaltyLSAcceptor](#) &)
Overloaded Equals Operator.

Private Attributes

- [Number](#) resto_pred_
When called from the restoration phase, this is the required predicted reduction.

Parameters for the penalty function line search

algorithm.

Names as in the filter paper

- [Number](#) nu_init_
Initial value of penalty parameter.
- [Number](#) nu_inc_
Increment for penalty parameter.

- [Number eta_](#)
 η_φ
- [Number rho_](#)
 ρ
- [Index max_soc_](#)
Maximal number of second order correction steps.
- [Number kappa_soc_](#)
Required reduction in constraint violation before trying multiple second order correction steps κ_{soc} .

Information related to watchdog procedure

- [Number reference_theta_](#)
Constraint violation at the point with respect to which progress is to be made.
- [Number reference_barr_](#)
Barrier objective function at the point with respect to which progress is to be made.
- [Number reference_gradBarrTDelta_](#)
Barrier gradient transpose search direction at the point with respect to which progress is to be made.
- [Number reference_dWd_](#)
Two-sided product of search direction with complete Hessian.
- [SmartPtr< const Vector > reference_JacC_delta_](#)
Product of Jacobian of equality constraint with x direction.
- [SmartPtr< const Vector > reference_JacD_delta_](#)
Product of Jacobian of (d-s) constraint with search direction.
- [Number reference_pred_](#)
Reference predicted reduction.
- [Number watchdog_theta_](#)
Constraint violation at reference point.
- [Number watchdog_barr_](#)
Barrier objective function at reference point.
- [Number watchdog_pred_](#)
Predicted reduction to be compared with in watch dog.

Penalty parameter

- [Number nu_](#)
Current value of the penalty parameter.
- [Number last_nu_](#)
Value of penalty parameter at beginning of the iteration.

Strategy objective that are used

- [SmartPtr< PDSystemSolver > pd_solver_](#)

Additional Inherited Members

6.130.1 Detailed Description

Penalty function line search.

This class implements the penalty function line search procedure as proposed by Waltz, Morales, Nocedal, Orban.

Definition at line 23 of file IpPenaltyLSAcceptor.hpp.

6.130.2 Constructor & Destructor Documentation

6.130.2.1 Ipopt::PenaltyLSAcceptor::PenaltyLSAcceptor (const SmartPtr< PDSystemSolver > & *pd_solver*)

Constructor.

The [PDSystemSolver](#) object only needs to be provided (i.e. not NULL) if second order correction or corrector steps are to be used.

6.130.2.2 virtual Ipopt::PenaltyLSAcceptor::~~PenaltyLSAcceptor () [virtual]

Default destructor.

6.130.2.3 Ipopt::PenaltyLSAcceptor::PenaltyLSAcceptor (const PenaltyLSAcceptor &) [private]

Copy Constructor.

6.130.3 Member Function Documentation

6.130.3.1 virtual bool Ipopt::PenaltyLSAcceptor::InitializeImpl (const OptionsList & *options*, const std::string & *prefix*) [virtual]

InitializeImpl - overloaded from [AlgorithmStrategyObject](#).

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.130.3.2 virtual void Ipopt::PenaltyLSAcceptor::Reset () [virtual]

Reset the acceptor.

This function should be called if all previous information should be discarded when the line search is performed the next time. For example, this method should be called if the barrier parameter is changed.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.130.3.3 virtual void Ipopt::PenaltyLSAcceptor::InitThisLineSearch (bool *in_watchdog*) [virtual]

Initialization for the next line search.

The flag *in_watchdog* indicates if we are currently in an active watchdog procedure.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.130.3.4 virtual void Ipopt::PenaltyLSAcceptor::PrepareRestoPhaseStart () [virtual]

Method that is called before the restoration phase is called.

Here, we can set up things that are required in the termination test for the restoration phase.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.130.3.5 virtual Number Ipopt::PenaltyLSAcceptor::CalculateAlphaMin () [virtual]

Method returning the lower bound on the trial step sizes.

Implements [Ipopt::BacktrackingLSAcceptor](#).

6.130.3.6 virtual bool Ipopt::PenaltyLSAcceptor::CheckAcceptabilityOfTrialPoint (Number *alpha_primal*) [virtual]

Method for checking if current trial point is acceptable.

It is assumed that the delta information in `ip_data` is the search direction used in criteria. The primal trial point has to be set before the call.

Implements [Ipopt::BacktrackingLSAcceptor](#).

```
6.130.3.7 virtual bool Ipopt::PenaltyLSAcceptor::TrySecondOrderCorrection ( Number alpha_primal_test, Number &
      alpha_primal, SmartPtr< IteratesVector > & actual_delta ) [virtual]
```

Try a second order correction for the constraints.

If the first trial step (with incoming `alpha_primal`) has been reject, this tries up to `max_soc` second order corrections for the constraints. Here, `alpha_primal_test` is the step size that has to be used in the penalty function acceptance tests. On output `actual_delta` has been set to the step including the second order correction if it has been accepted, otherwise it is unchanged. If the SOC step has been accepted, `alpha_primal` has the fraction-to-the-boundary value for the SOC step on output. The return value is true, if a SOC step has been accepted.

Implements [Ipopt::BacktrackingLSAcceptor](#).

```
6.130.3.8 virtual bool Ipopt::PenaltyLSAcceptor::TryCorrector ( Number alpha_primal_test, Number & alpha_primal,
      SmartPtr< IteratesVector > & actual_delta ) [virtual]
```

Try higher order corrector (for fast local convergence).

In contrast to a second order correction step, which tries to make an unacceptable point acceptable by improving constraint violation, this corrector step is tried even if the regular primal-dual step is acceptable.

Implements [Ipopt::BacktrackingLSAcceptor](#).

```
6.130.3.9 virtual char Ipopt::PenaltyLSAcceptor::UpdateForNextIteration ( Number alpha_primal_test ) [virtual]
```

Method for ending the current line search.

When it is called, the internal data should be updates. `alpha_primal_test` is the value of alpha that has been used for in the acceptance test ealier.

Implements [Ipopt::BacktrackingLSAcceptor](#).

```
6.130.3.10 virtual void Ipopt::PenaltyLSAcceptor::StartWatchDog ( ) [virtual]
```

Method for setting internal data if the watchdog procedure is started.

Implements [Ipopt::BacktrackingLSAcceptor](#).

```
6.130.3.11 virtual void Ipopt::PenaltyLSAcceptor::StopWatchDog ( ) [virtual]
```

Method for setting internal data if the watchdog procedure is stopped.

Implements [Ipopt::BacktrackingLSAcceptor](#).

```
6.130.3.12 bool Ipopt::PenaltyLSAcceptor::IsAcceptableToCurrentIterate ( Number trial_barr, Number trial_theta, bool
      called_from_restoration = false ) const
```

Checks if a trial point is acceptable to the current iterate.

```
6.130.3.13 static void Ipopt::PenaltyLSAcceptor::RegisterOptions ( SmartPtr< RegisteredOptions > options )
      [static]
```

Methods for [OptionsList](#).

6.130.3.14 `void Ipopt::PenaltyLSAcceptor::operator= (const PenaltyLSAcceptor &) [private]`

Overloaded Equals Operator.

6.130.3.15 `Number Ipopt::PenaltyLSAcceptor::CalcPred (Number alpha) [private]`

Compute predicted reduction for given step size.

6.130.4 Member Data Documentation

6.130.4.1 `Number Ipopt::PenaltyLSAcceptor::nu_init_ [private]`

Initial value of penalty parameter.

Definition at line 147 of file IpPenaltyLSAcceptor.hpp.

6.130.4.2 `Number Ipopt::PenaltyLSAcceptor::nu_inc_ [private]`

Increment for penalty parameter.

Definition at line 149 of file IpPenaltyLSAcceptor.hpp.

6.130.4.3 `Number Ipopt::PenaltyLSAcceptor::eta_ [private]`

η_ϕ

Definition at line 151 of file IpPenaltyLSAcceptor.hpp.

6.130.4.4 `Number Ipopt::PenaltyLSAcceptor::rho_ [private]`

ρ

Definition at line 153 of file IpPenaltyLSAcceptor.hpp.

6.130.4.5 `Index Ipopt::PenaltyLSAcceptor::max_soc_ [private]`

Maximal number of second order correction steps.

Definition at line 155 of file IpPenaltyLSAcceptor.hpp.

6.130.4.6 `Number Ipopt::PenaltyLSAcceptor::kappa_soc_ [private]`

Required reduction in constraint violation before trying multiple second order correction steps κ_{soc} .

Definition at line 159 of file IpPenaltyLSAcceptor.hpp.

6.130.4.7 `Number Ipopt::PenaltyLSAcceptor::reference_theta_ [private]`

Constraint violation at the point with respect to which progress is to be made.

Definition at line 166 of file IpPenaltyLSAcceptor.hpp.

6.130.4.8 `Number Ipopt::PenaltyLSAcceptor::reference_barr_ [private]`

Barrier objective function at the point with respect to which progress is to be made.

Definition at line 169 of file IpPenaltyLSAcceptor.hpp.

6.130.4.9 Number Ipopt::PenaltyLSAcceptor::reference_gradBarrTDelta_ [private]

Barrier gradient transpose search direction at the point with respect to which progress is to be made.

Definition at line 172 of file IpPenaltyLSAcceptor.hpp.

6.130.4.10 Number Ipopt::PenaltyLSAcceptor::reference_dWd_ [private]

Two-sided product of search direction with complete Hessian.

Definition at line 174 of file IpPenaltyLSAcceptor.hpp.

6.130.4.11 SmartPtr<const Vector> Ipopt::PenaltyLSAcceptor::reference_JacC_delta_ [private]

Product of Jacobian of equality constraint with x direction.

Definition at line 176 of file IpPenaltyLSAcceptor.hpp.

6.130.4.12 SmartPtr<const Vector> Ipopt::PenaltyLSAcceptor::reference_JacD_delta_ [private]

Product of Jacobian of (d-s) constraint with search direction.

Definition at line 178 of file IpPenaltyLSAcceptor.hpp.

6.130.4.13 Number Ipopt::PenaltyLSAcceptor::reference_pred_ [private]

Reference predicted reduction.

If positive, then it is used in watch dog.

Definition at line 181 of file IpPenaltyLSAcceptor.hpp.

6.130.4.14 Number Ipopt::PenaltyLSAcceptor::watchdog_theta_ [private]

Constraint violation at reference point.

Definition at line 183 of file IpPenaltyLSAcceptor.hpp.

6.130.4.15 Number Ipopt::PenaltyLSAcceptor::watchdog_barr_ [private]

Barrier objective function at reference point.

Definition at line 185 of file IpPenaltyLSAcceptor.hpp.

6.130.4.16 Number Ipopt::PenaltyLSAcceptor::watchdog_pred_ [private]

Predicted reduction to be compared with in watch dog.

Definition at line 187 of file IpPenaltyLSAcceptor.hpp.

6.130.4.17 Number Ipopt::PenaltyLSAcceptor::nu_ [private]

Current value of the penalty parameter.

Definition at line 193 of file IpPenaltyLSAcceptor.hpp.

6.130.4.18 Number Ipopt::PenaltyLSAcceptor::last_nu_ [private]

Value of penalty parameter at beginning of the iteration.

Definition at line 195 of file IpPenaltyLSAcceptor.hpp.

6.130.4.19 Number Ipopt::PenaltyLSAcceptor::resto_pred_ [private]

When called from the restoration phase, this is the required predicted reduction.

Definition at line 200 of file IpPenaltyLSAcceptor.hpp.

6.130.4.20 SmartPtr<PDSolver> Ipopt::PenaltyLSAcceptor::pd_solver_ [private]

Definition at line 204 of file IpPenaltyLSAcceptor.hpp.

The documentation for this class was generated from the following file:

- [Algorithm/IpPenaltyLSAcceptor.hpp](#)

6.131 Ipopt::PiecewisePenalty Class Reference

Class for the Piecewise Penalty.

```
#include <IpPiecewisePenalty.hpp>
```

Public Member Functions

- void [Clear](#) ()
Delete all Piecewise Penalty entries.
- void [Print](#) (const [Journalist](#) &jnlst)
Print current Piecewise Penalty entries.

Constructors/Destructors

- [PiecewisePenalty](#) ([Index](#) dim)
Default Constructor.
- [~PiecewisePenalty](#) ()
Default Destructor.
- bool [IsPiecewisePenaltyListEmpty](#) ()
- void [InitPiecewisePenaltyList](#) ([Number](#) pen_r, [Number](#) barrier_obj, [Number](#) infeasi)
- bool [Acceptable](#) ([Number](#) Fzconst, [Number](#) Fzlin)
Check acceptability of given coordinates with respect to the Piecewise Penalty.
- [Number](#) [BiggestBarr](#) ()
Get the value of the biggest barrier function so far.
- void [UpdateEntry](#) ([Number](#) barrier_obj, [Number](#) infeasi)
Update Piecewise Penalty entry for given coordinates.
- void [AddEntry](#) ([Number](#) pen_r, [Number](#) barrier_obj, [Number](#) infeasi)
Add a entry to the list.
- void [ResetList](#) ([Number](#) pen_r, [Number](#) barrier_obj, [Number](#) infeasi)
Clear and reset the piecewise penalty list.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `PiecewisePenalty ()`
Default Constructor.
- `PiecewisePenalty (const PiecewisePenalty &)`
Copy Constructor.
- `void operator= (const PiecewisePenalty &)`
Overloaded Equals Operator.

Private Attributes

- `Index dim_`
Dimension of the Piecewise Penalty (number of coordinates per entry)
- `Number min_piece_penalty_`
The min penalty value for the piecewise penalty list.
- `Index max_piece_number_`
The max number of the break points in the piecewise penalty list.
- `std::vector< PiecewisePenEntry > PiecewisePenalty_list_`
vector storing the Piecewise Penalty entries

6.131.1 Detailed Description

Class for the Piecewise Penalty.

This class contains all Piecewise Penalty entries. The entries are stored as the corner point, including the margin.

Definition at line 39 of file `IpPiecewisePenalty.hpp`.

6.131.2 Constructor & Destructor Documentation

6.131.2.1 `Ipopt::PiecewisePenalty::PiecewisePenalty (Index dim)`

Default Constructor.

6.131.2.2 `Ipopt::PiecewisePenalty::~~PiecewisePenalty () [inline]`

Default Destructor.

Definition at line 47 of file `IpPiecewisePenalty.hpp`.

6.131.2.3 `Ipopt::PiecewisePenalty::PiecewisePenalty () [private]`

Default Constructor.

6.131.2.4 `Ipopt::PiecewisePenalty::PiecewisePenalty (const PiecewisePenalty &) [private]`

Copy Constructor.

6.131.3 Member Function Documentation

6.131.3.1 **bool** Ipopt::PiecewisePenalty::IsPiecewisePenaltyListEmpty () [inline]

Definition at line 57 of file IpPiecewisePenalty.hpp.

6.131.3.2 **void** Ipopt::PiecewisePenalty::InitPiecewisePenaltyList (**Number** *pen_r*, **Number** *barrier_obj*, **Number** *infeasi*) [inline]

Definition at line 62 of file IpPiecewisePenalty.hpp.

6.131.3.3 **bool** Ipopt::PiecewisePenalty::Acceptable (**Number** *Fzconst*, **Number** *Fzlin*)

Check acceptability of given coordinates with respect to the Piecewise Penalty.

Returns true, if pair is acceptable

6.131.3.4 **Number** Ipopt::PiecewisePenalty::BiggestBarr ()

Get the value of the biggest barrier function so far.

6.131.3.5 **void** Ipopt::PiecewisePenalty::UpdateEntry (**Number** *barrier_obj*, **Number** *infeasi*)

Update Piecewise Penalty entry for given coordinates.

6.131.3.6 **void** Ipopt::PiecewisePenalty::AddEntry (**Number** *pen_r*, **Number** *barrier_obj*, **Number** *infeasi*) [inline]

Add a entry to the list.

Definition at line 81 of file IpPiecewisePenalty.hpp.

6.131.3.7 **void** Ipopt::PiecewisePenalty::ResetList (**Number** *pen_r*, **Number** *barrier_obj*, **Number** *infeasi*) [inline]

Clear and reset the piecewise penalty list.

Definition at line 97 of file IpPiecewisePenalty.hpp.

6.131.3.8 **void** Ipopt::PiecewisePenalty::Clear () [inline]

Delete all Piecewise Penalty entries.

Definition at line 106 of file IpPiecewisePenalty.hpp.

6.131.3.9 **void** Ipopt::PiecewisePenalty::Print (**const** **Journalist** & *jnlst*)

Print current Piecewise Penalty entries.

6.131.3.10 **void** Ipopt::PiecewisePenalty::operator= (**const** **PiecewisePenalty** &) [private]

Overloaded Equals Operator.

6.131.4 Member Data Documentation

6.131.4.1 **Index** Ipopt::PiecewisePenalty::dim_ [private]

Dimension of the Piecewise Penalty (number of coordinates per entry)

Definition at line 133 of file IpPiecewisePenalty.hpp.

6.131.4.2 **Number** Ipopt::PiecewisePenalty::min_piece_penalty_ [private]

The min penalty value for the piecewise penalty list.

Definition at line 136 of file IpPiecewisePenalty.hpp.

6.131.4.3 **Index** Ipopt::PiecewisePenalty::max_piece_number_ [private]

The max number of the break points in the piecewise penalty list.

Definition at line 139 of file IpPiecewisePenalty.hpp.

6.131.4.4 **std::vector<PiecewisePenEntry>** Ipopt::PiecewisePenalty::PiecewisePenalty_list_ [private]

vector storing the Piecewise Penalty entries

Definition at line 142 of file IpPiecewisePenalty.hpp.

The documentation for this class was generated from the following file:

- contrib/CGPenalty/[IpPiecewisePenalty.hpp](#)

6.132 Ipopt::PiecewisePenEntry Struct Reference

struct for one Piecewise Penalty entry.

```
#include <IpPiecewisePenalty.hpp>
```

Public Attributes

- [Number](#) pen_r
- [Number](#) barrier_obj
- [Number](#) infeas_i

6.132.1 Detailed Description

struct for one Piecewise Penalty entry.

Definition at line 25 of file IpPiecewisePenalty.hpp.

6.132.2 Member Data Documentation

6.132.2.1 **Number** Ipopt::PiecewisePenEntry::pen_r

Definition at line 28 of file IpPiecewisePenalty.hpp.

6.132.2.2 **Number** Ipopt::PiecewisePenEntry::barrier_obj

Definition at line 29 of file IpPiecewisePenalty.hpp.

6.132.2.3 **Number** Ipopt::PiecewisePenEntry::infeas_i

Definition at line 30 of file IpPiecewisePenalty.hpp.

The documentation for this struct was generated from the following file:

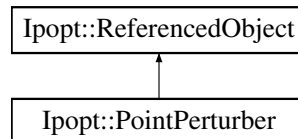
- [contrib/CGPenalty/IpPiecewisePenalty.hpp](#)

6.133 Ipopt::PointPerturber Class Reference

This class is a simple object for generating randomly perturbed points that are within the [NLP](#) bounds.

```
#include <IpEquilibrationScaling.hpp>
```

Inheritance diagram for Ipopt::PointPerturber:



Public Member Functions

- [SmartPointer< Vector > MakeNewPerturbedPoint \(\)](#) const

Return a new perturbed point.

Constructors/Destructors

- [PointPerturber](#) (const [Vector](#) &reference_point, [Number](#) random_pert_radius, const [Matrix](#) &Px_L, const [Vector](#) &x_L, const [Matrix](#) &Px_U, const [Vector](#) &x_U)
- virtual [~PointPerturber](#) ()

Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [PointPerturber](#) (const [PointPerturber](#) &)
 - void [operator=](#) (const [PointPerturber](#) &)
- Copy Constructor.*
Overloaded Equals Operator.

Private Attributes

- [SmartPointer< Vector > ref_point_](#)
pointer to the midpoint of the perturbation
- [SmartPointer< Vector > pert_dir_](#)
pointer to the perturbation vector

6.133.1 Detailed Description

This class is a simple object for generating randomly perturbed points that are within the [NLP](#) bounds.

The `random_perturb_radius` gives the upper bound of the perturbation.

Definition at line 92 of file `IpEquilibrationScaling.hpp`.

6.133.2 Constructor & Destructor Documentation

6.133.2.1 `Ipopt::PointPerturber::PointPerturber (const Vector & reference_point, Number random_pert_radius, const Matrix & Px_L, const Vector & x_L, const Matrix & Px_U, const Vector & x_U)`

6.133.2.2 `virtual Ipopt::PointPerturber::~~PointPerturber () [inline], [virtual]`

Default destructor.

Definition at line 103 of file `IpEquilibrationScaling.hpp`.

6.133.2.3 `Ipopt::PointPerturber::PointPerturber (const PointPerturber &) [private]`

Copy Constructor.

6.133.3 Member Function Documentation

6.133.3.1 `SmartPtr<Vector> Ipopt::PointPerturber::MakeNewPerturbedPoint () const`

Return a new perturbed point.

6.133.3.2 `void Ipopt::PointPerturber::operator= (const PointPerturber &) [private]`

Overloaded Equals Operator.

6.133.4 Member Data Documentation

6.133.4.1 `SmartPtr<Vector> Ipopt::PointPerturber::ref_point_ [private]`

pointer to the midpoint of the perturbation

Definition at line 129 of file `IpEquilibrationScaling.hpp`.

6.133.4.2 `SmartPtr<Vector> Ipopt::PointPerturber::pert_dir_ [private]`

pointer to the perturbation vector

Definition at line 132 of file `IpEquilibrationScaling.hpp`.

The documentation for this class was generated from the following file:

- [Algorithm/IpEquilibrationScaling.hpp](#)

6.134 Ipopt::AmplOptionsList::PrivatInfo Class Reference

```
#include <AmplTNLP.hpp>
```

Public Member Functions

- **PrivatInfo** (const std::string ipopt_name, [SmartPtr< OptionsList >](#) options, [SmartPtr< const Journalist >](#) jnlst, void **nerror=NULL)
- const std::string & [IpoptName](#) () const
- const [SmartPtr< OptionsList >](#) & [Options](#) () const
- const [SmartPtr< const Journalist >](#) & [Jnlst](#) () const
- void ** [NError](#) ()

Private Attributes

- const std::string [ipopt_name_](#)
- const [SmartPtr< OptionsList >](#) [options_](#)
- const [SmartPtr< const Journalist >](#) [jnlst_](#)
- void ** [nerror_](#)

6.134.1 Detailed Description

Definition at line 163 of file AmplTNLP.hpp.

6.134.2 Constructor & Destructor Documentation

- 6.134.2.1 **Ipopt::AmplOptionsList::PrivatInfo::PrivatInfo** (const std::string *ipopt_name*, [SmartPtr< OptionsList >](#) *options*, [SmartPtr< const Journalist >](#) *jnlst*, void ** *nerror* =NULL) [\[inline\]](#)

Definition at line 166 of file AmplTNLP.hpp.

6.134.3 Member Function Documentation

- 6.134.3.1 const std::string& **Ipopt::AmplOptionsList::PrivatInfo::IpoptName** () const [\[inline\]](#)

Definition at line 176 of file AmplTNLP.hpp.

- 6.134.3.2 const [SmartPtr<OptionsList>](#)& **Ipopt::AmplOptionsList::PrivatInfo::Options** () const [\[inline\]](#)

Definition at line 180 of file AmplTNLP.hpp.

- 6.134.3.3 const [SmartPtr<const Journalist>](#)& **Ipopt::AmplOptionsList::PrivatInfo::Jnlst** () const [\[inline\]](#)

Definition at line 184 of file AmplTNLP.hpp.

- 6.134.3.4 void** **Ipopt::AmplOptionsList::PrivatInfo::NError** () [\[inline\]](#)

Definition at line 188 of file AmplTNLP.hpp.

6.134.4 Member Data Documentation

- 6.134.4.1 const std::string **Ipopt::AmplOptionsList::PrivatInfo::ipopt_name_** [\[private\]](#)

Definition at line 193 of file AmplTNLP.hpp.

6.134.4.2 `const SmartPtr<OptionsList> Ipopt::AmplOptionsList::PrivatInfo::options_ [private]`

Definition at line 194 of file AmplTNLP.hpp.

6.134.4.3 `const SmartPtr<const Journalist> Ipopt::AmplOptionsList::PrivatInfo::jnlst_ [private]`

Definition at line 195 of file AmplTNLP.hpp.

6.134.4.4 `void** Ipopt::AmplOptionsList::PrivatInfo::nerror_ [private]`

Definition at line 196 of file AmplTNLP.hpp.

The documentation for this class was generated from the following file:

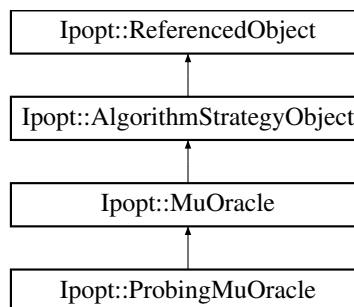
- Apps/AmplSolver/[AmplTNLP.hpp](#)

6.135 Ipopt::ProbingMuOracle Class Reference

Implementation of the probing strategy for computing the barrier parameter.

```
#include <IpProbingMuOracle.hpp>
```

Inheritance diagram for Ipopt::ProbingMuOracle:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [CalculateMu](#) ([Number](#) mu_min, [Number](#) mu_max, [Number](#) &new_mu)
Method for computing the value of the barrier parameter that could be used in the current iteration (using Mehrotra's probing heuristic).

Constructors/Destructors

- [ProbingMuOracle](#) (const [SmartPtr](#)< [PDSolver](#) > &pd_solver)
Constructor.
- virtual [~ProbingMuOracle](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

- [Number CalculateAffineMu](#) ([Number](#) alpha_primal, [Number](#) alpha_dual, const [IteratesVector](#) &step)
Auxilliary function for computing the average complementarity at a point, given step sizes and step.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ProbingMuOracle](#) ()
Default Constructor.
- [ProbingMuOracle](#) (const [ProbingMuOracle](#) &)
Copy Constructor.
- void [operator=](#) (const [ProbingMuOracle](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [PDSystemSolver](#) > [pd_solver_](#)
Pointer to the object that should be used to solve the primal-dual system.

Algorithmic parameters

- [Number sigma_max_](#)
safeguarding upper bound on centering parameter sigma

Additional Inherited Members

6.135.1 Detailed Description

Implementation of the probing strategy for computing the barrier parameter.

Definition at line 21 of file IpProbingMuOracle.hpp.

6.135.2 Constructor & Destructor Documentation

6.135.2.1 `Ipopt::ProbingMuOracle::ProbingMuOracle (const SmartPtr< PDSystemSolver > &pd_solver)`

Constructor.

6.135.2.2 `virtual Ipopt::ProbingMuOracle::~~ProbingMuOracle () [virtual]`

Default destructor.

6.135.2.3 `Ipopt::ProbingMuOracle::ProbingMuOracle () [private]`

Default Constructor.

6.135.2.4 `Ipopt::ProbingMuOracle::ProbingMuOracle (const ProbingMuOracle &) [private]`

Copy Constructor.

6.135.3 Member Function Documentation

6.135.3.1 `virtual bool Ipopt::ProbingMuOracle::InitializeImpl (const OptionsList & options, const std::string & prefix)`
[virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::MuOracle](#).

6.135.3.2 `virtual bool Ipopt::ProbingMuOracle::CalculateMu (Number mu_min, Number mu_max, Number & new_mu)`
[virtual]

Method for computing the value of the barrier parameter that could be used in the current iteration (using Mehrotra's probing heuristic).

Implements [Ipopt::MuOracle](#).

6.135.3.3 `static void Ipopt::ProbingMuOracle::RegisterOptions (SmartPtr< RegisteredOptions > roptions)` [static]

Methods for IpoptType.

6.135.3.4 `void Ipopt::ProbingMuOracle::operator= (const ProbingMuOracle &)` [private]

Overloaded Equals Operator.

6.135.3.5 `Number Ipopt::ProbingMuOracle::CalculateAffineMu (Number alpha_primal, Number alpha_dual, const IteratesVector & step)` [private]

Auxilliary function for computing the average complementarity at a point, given step sizes and step.

6.135.4 Member Data Documentation

6.135.4.1 `SmartPtr<PDSysolver> Ipopt::ProbingMuOracle::pd_solver_` [private]

Pointer to the object that should be used to solve the primal-dual system.

Definition at line 69 of file IpProbingMuOracle.hpp.

6.135.4.2 `Number Ipopt::ProbingMuOracle::sigma_max_` [private]

safeguarding upper bound on centering parameter sigma

Definition at line 81 of file IpProbingMuOracle.hpp.

The documentation for this class was generated from the following file:

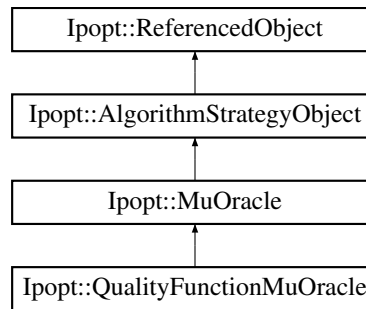
- [Algorithm/IpProbingMuOracle.hpp](#)

6.136 Ipopt::QualityFunctionMuOracle Class Reference

Implementation of the probing strategy for computing the barrier parameter.

`#include <IpQualityFunctionMuOracle.hpp>`

Inheritance diagram for Ipopt::QualityFunctionMuOracle:



Public Types

Public enums. Some of those are also used for the
quality function

- enum [NormEnum](#) { [NM_NORM_1](#) =0, [NM_NORM_2_SQUARED](#), [NM_NORM_MAX](#), [NM_NORM_2](#) }
enum for norm type
- enum [CentralityEnum](#) { [CEN_NONE](#) =0, [CEN_LOG](#), [CEN_RECIPROCAL](#), [CEN_CUBED_RECIPROCAL](#) }
enum for centrality type
- enum [BalancingTermEnum](#) { [BT_NONE](#) =0, [BT_CUBIC](#) }
enum for the quality function balancing term type

Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [CalculateMu](#) ([Number](#) mu_min, [Number](#) mu_max, [Number](#) &new_mu)
Method for computing the value of the barrier parameter that could be used in the current iteration (using the LOQO formula).

Constructors/Destructors

- [QualityFunctionMuOracle](#) (const [SmartPtr](#)< [PDSysstemSolver](#) > &pd_solver)
Constructor.
- virtual [~QualityFunctionMuOracle](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

- [Number](#) [CalculateQualityFunction](#) ([Number](#) sigma, const [Vector](#) &step_aff_x_L, const [Vector](#) &step_aff_x_U, const [Vector](#) &step_aff_s_L, const [Vector](#) &step_aff_s_U, const [Vector](#) &step_aff_y_c, const [Vector](#) &step_aff_y_d, const [Vector](#) &step_aff_z_L, const [Vector](#) &step_aff_z_U, const [Vector](#) &step_aff_v_L, const [Vector](#) &step_aff_v_U, const [Vector](#) &step_cen_x_L, const [Vector](#) &step_cen_x_U, const [Vector](#) &step_cen_s_L, const [Vector](#) &step_cen_s_U, const [Vector](#) &step_cen_y_c, const [Vector](#) &step_cen_y_d, const [Vector](#) &step_cen_z_L, const [Vector](#) &step_cen_z_U, const [Vector](#) &step_cen_v_L, const [Vector](#) &step_cen_v_U)

Auxilliary function for computing the average complementarity at a point, given step sizes and step.

- [Number PerformGoldenSection](#) ([Number](#) sigma_up, [Number](#) q_up, [Number](#) sigma_lo, [Number](#) q_lo, [Number](#) sigma_tol, [Number](#) qf_tol, const [Vector](#) &step_aff_x_L, const [Vector](#) &step_aff_x_U, const [Vector](#) &step_aff_s_L, const [Vector](#) &step_aff_s_U, const [Vector](#) &step_aff_y_c, const [Vector](#) &step_aff_y_d, const [Vector](#) &step_aff_z_L, const [Vector](#) &step_aff_z_U, const [Vector](#) &step_aff_v_L, const [Vector](#) &step_aff_v_U, const [Vector](#) &step_cen_x_L, const [Vector](#) &step_cen_x_U, const [Vector](#) &step_cen_s_L, const [Vector](#) &step_cen_s_U, const [Vector](#) &step_cen_y_c, const [Vector](#) &step_cen_y_d, const [Vector](#) &step_cen_z_L, const [Vector](#) &step_cen_z_U, const [Vector](#) &step_cen_v_L, const [Vector](#) &step_cen_v_U)

Auxilliary function performing the golden section.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [QualityFunctionMuOracle](#) ()
Default Constructor.
- [QualityFunctionMuOracle](#) (const [QualityFunctionMuOracle](#) &)
Copy Constructor.
- void [operator=](#) (const [QualityFunctionMuOracle](#) &)
Overloaded Equals Operator.
- [Number ScaleSigma](#) ([Number](#) sigma)
Auxilliary functions for scaling the sigma axis in the golden section procedure.
- [Number UnscaleSigma](#) ([Number](#) scaled_sigma)

Private Attributes

- [SmartPtr](#)< [PDSysstemSolver](#) > [pd_solver_](#)
Pointer to the object that should be used to solve the primal-dual system.
- [Index](#) [count_qf_evals_](#)

Algorithmic parameters

Auxilliary function performing the golden section in the logarithmic scale

- [Number](#) [sigma_max_](#)
Upper bound on centering parameter sigma.
- [Number](#) [sigma_min_](#)
Lower bound on centering parameter sigma.
- [NormEnum](#) [quality_function_norm_](#)
Norm to be used for the quality function.
- [CentralityEnum](#) [quality_function_centrality_](#)
Flag indicating how centrality should be involved in the quality function.
- [BalancingTermEnum](#) [quality_function_balancing_term_](#)
Flag indicating whether we use a balancing term in the quality function.
- [Number](#) [quality_function_section_sigma_tol_](#)
Relative tolerance for golden bi-section algorithm in sigma space.
- [Number](#) [quality_function_section_qf_tol_](#)
Relative tolerance for golden bi-section algorithm in function value space.
- [Index](#) [quality_function_max_section_steps_](#)
Maximal number of bi-section steps in the golden section search for sigma.

Temporary work space vectors. We use those to avoid

repeated reallocation in CalculateQualityFunction.

- SmartPtr< Vector > tmp_step_x_L_
- SmartPtr< Vector > tmp_step_x_U_
- SmartPtr< Vector > tmp_step_s_L_
- SmartPtr< Vector > tmp_step_s_U_
- SmartPtr< Vector > tmp_step_z_L_
- SmartPtr< Vector > tmp_step_z_U_
- SmartPtr< Vector > tmp_step_v_L_
- SmartPtr< Vector > tmp_step_v_U_
- SmartPtr< Vector > tmp_slack_x_L_
- SmartPtr< Vector > tmp_slack_x_U_
- SmartPtr< Vector > tmp_slack_s_L_
- SmartPtr< Vector > tmp_slack_s_U_
- SmartPtr< Vector > tmp_z_L_
- SmartPtr< Vector > tmp_z_U_
- SmartPtr< Vector > tmp_v_L_
- SmartPtr< Vector > tmp_v_U_

Quantities used many times in CalculateQualityFunction,

which we store here instead of retrieving them from cache every time.

I (AW) don't know if that really makes a difference, but some of those things showed up in gprof.

- bool initialized_
- Index n_dual_
- Index n_pri_
- Index n_comp_
- SmartPtr< const Vector > curr_slack_x_L_
- SmartPtr< const Vector > curr_slack_x_U_
- SmartPtr< const Vector > curr_slack_s_L_
- SmartPtr< const Vector > curr_slack_s_U_
- SmartPtr< const Vector > curr_z_L_
- SmartPtr< const Vector > curr_z_U_
- SmartPtr< const Vector > curr_v_L_
- SmartPtr< const Vector > curr_v_U_
- Number curr_grad_lag_x_asum_
- Number curr_grad_lag_s_asum_
- Number curr_c_asum_
- Number curr_d_minus_s_asum_
- Number curr_grad_lag_x_nrm2_
- Number curr_grad_lag_s_nrm2_
- Number curr_c_nrm2_
- Number curr_d_minus_s_nrm2_
- Number curr_grad_lag_x_amax_
- Number curr_grad_lag_s_amax_
- Number curr_c_amax_
- Number curr_d_minus_s_amax_

Additional Inherited Members**6.136.1 Detailed Description**

Implementation of the probing strategy for computing the barrier parameter.

Definition at line 22 of file IpQualityFunctionMuOracle.hpp.

6.136.2 Member Enumeration Documentation

6.136.2.1 enum `Ipopt::QualityFunctionMuOracle::NormEnum`

enum for norm type

Enumerator

NM_NORM_1
NM_NORM_2_SQUARED
NM_NORM_MAX
NM_NORM_2

Definition at line 51 of file `IpQualityFunctionMuOracle.hpp`.

6.136.2.2 enum `Ipopt::QualityFunctionMuOracle::CentralityEnum`

enum for centrality type

Enumerator

CEN_NONE
CEN_LOG
CEN_RECIPROCAL
CEN_CUBED_RECIPROCAL

Definition at line 59 of file `IpQualityFunctionMuOracle.hpp`.

6.136.2.3 enum `Ipopt::QualityFunctionMuOracle::BalancingTermEnum`

enum for the quality function balancing term type

Enumerator

BT_NONE
BT_CUBIC

Definition at line 67 of file `IpQualityFunctionMuOracle.hpp`.

6.136.3 Constructor & Destructor Documentation

6.136.3.1 `Ipopt::QualityFunctionMuOracle::QualityFunctionMuOracle (const SmartPtr< PDSystemSolver > & pd_solver)`

Constructor.

6.136.3.2 `virtual Ipopt::QualityFunctionMuOracle::~~QualityFunctionMuOracle () [virtual]`

Default destructor.

6.136.3.3 `Ipopt::QualityFunctionMuOracle::QualityFunctionMuOracle () [private]`

Default Constructor.

6.136.3.4 `Ipopt::QualityFunctionMuOracle::QualityFunctionMuOracle (const QualityFunctionMuOracle &) [private]`

Copy Constructor.

6.136.4 Member Function Documentation

6.136.4.1 `virtual bool Ipopt::QualityFunctionMuOracle::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::MuOracle](#).

6.136.4.2 `virtual bool Ipopt::QualityFunctionMuOracle::CalculateMu (Number mu_min, Number mu_max, Number & new_mu) [virtual]`

Method for computing the value of the barrier parameter that could be used in the current iteration (using the LOQO formula).

Implements [Ipopt::MuOracle](#).

6.136.4.3 `static void Ipopt::QualityFunctionMuOracle::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for IpoptType.

6.136.4.4 `void Ipopt::QualityFunctionMuOracle::operator= (const QualityFunctionMuOracle &) [private]`

Overloaded Equals Operator.

6.136.4.5 `Number Ipopt::QualityFunctionMuOracle::CalculateQualityFunction (Number sigma, const Vector & step_aff_x_L, const Vector & step_aff_x_U, const Vector & step_aff_s_L, const Vector & step_aff_s_U, const Vector & step_aff_y_c, const Vector & step_aff_y_d, const Vector & step_aff_z_L, const Vector & step_aff_z_U, const Vector & step_aff_v_L, const Vector & step_aff_v_U, const Vector & step_cen_x_L, const Vector & step_cen_x_U, const Vector & step_cen_s_L, const Vector & step_cen_s_U, const Vector & step_cen_y_c, const Vector & step_cen_y_d, const Vector & step_cen_z_L, const Vector & step_cen_z_U, const Vector & step_cen_v_L, const Vector & step_cen_v_U) [private]`

Auxilliary function for computing the average complementarity at a point, given step sizes and step.

6.136.4.6 `Number Ipopt::QualityFunctionMuOracle::PerformGoldenSection (Number sigma_up, Number q_up, Number sigma_lo, Number q_lo, Number sigma_tol, Number qf_tol, const Vector & step_aff_x_L, const Vector & step_aff_x_U, const Vector & step_aff_s_L, const Vector & step_aff_s_U, const Vector & step_aff_y_c, const Vector & step_aff_y_d, const Vector & step_aff_z_L, const Vector & step_aff_z_U, const Vector & step_aff_v_L, const Vector & step_aff_v_U, const Vector & step_cen_x_L, const Vector & step_cen_x_U, const Vector & step_cen_s_L, const Vector & step_cen_s_U, const Vector & step_cen_y_c, const Vector & step_cen_y_d, const Vector & step_cen_z_L, const Vector & step_cen_z_U, const Vector & step_cen_v_L, const Vector & step_cen_v_U) [private]`

Auxilliary function performing the golden section.

6.136.4.7 `Number Ipopt::QualityFunctionMuOracle::ScaleSigma (Number sigma) [private]`

Auxilliary functions for scaling the sigma axis in the golden section procedure.

6.136.4.8 `Number Ipopt::QualityFunctionMuOracle::UnscaleSigma (Number scaled_sigma) [private]`

6.136.5 Member Data Documentation

6.136.5.1 **SmartPtr<PDSys...>** `Ipopt::QualityFunctionMuOracle::pd_solver_` [private]

Pointer to the object that should be used to solve the primal-dual system.

Definition at line 96 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.2 **Number** `Ipopt::QualityFunctionMuOracle::sigma_max_` [private]

Upper bound on centering parameter sigma.

Definition at line 190 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.3 **Number** `Ipopt::QualityFunctionMuOracle::sigma_min_` [private]

Lower bound on centering parameter sigma.

Definition at line 192 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.4 **NormEnum** `Ipopt::QualityFunctionMuOracle::quality_function_norm_` [private]

Norm to be used for the quality function.

Definition at line 194 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.5 **CentralityEnum** `Ipopt::QualityFunctionMuOracle::quality_function_centrality_` [private]

Flag indicating how centrality should be involved in the quality function.

Definition at line 197 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.6 **BalancingTermEnum** `Ipopt::QualityFunctionMuOracle::quality_function_balancing_term_` [private]

Flag indicating whether we use a balancing term in the quality function.

Definition at line 201 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.7 **Number** `Ipopt::QualityFunctionMuOracle::quality_function_section_sigma_tol_` [private]

Relative tolerance for golden bi-section algorithm in sigma space.

Definition at line 204 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.8 **Number** `Ipopt::QualityFunctionMuOracle::quality_function_section_qf_tol_` [private]

Relative tolerance for golden bi-section algorithm in function value space.

Definition at line 207 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.9 **Index** `Ipopt::QualityFunctionMuOracle::quality_function_max_section_steps_` [private]

Maximal number of bi-section steps in the golden section search for sigma.

Definition at line 210 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.10 **SmartPtr<Vector>** `Ipopt::QualityFunctionMuOracle::tmp_step_x_L_` [private]

Definition at line 216 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.11 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_step_x_U_ [private]

Definition at line 217 of file IpQualityFunctionMuOracle.hpp.

6.136.5.12 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_step_s_L_ [private]

Definition at line 218 of file IpQualityFunctionMuOracle.hpp.

6.136.5.13 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_step_s_U_ [private]

Definition at line 219 of file IpQualityFunctionMuOracle.hpp.

6.136.5.14 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_step_z_L_ [private]

Definition at line 220 of file IpQualityFunctionMuOracle.hpp.

6.136.5.15 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_step_z_U_ [private]

Definition at line 221 of file IpQualityFunctionMuOracle.hpp.

6.136.5.16 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_step_v_L_ [private]

Definition at line 222 of file IpQualityFunctionMuOracle.hpp.

6.136.5.17 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_step_v_U_ [private]

Definition at line 223 of file IpQualityFunctionMuOracle.hpp.

6.136.5.18 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_slack_x_L_ [private]

Definition at line 225 of file IpQualityFunctionMuOracle.hpp.

6.136.5.19 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_slack_x_U_ [private]

Definition at line 226 of file IpQualityFunctionMuOracle.hpp.

6.136.5.20 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_slack_s_L_ [private]

Definition at line 227 of file IpQualityFunctionMuOracle.hpp.

6.136.5.21 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_slack_s_U_ [private]

Definition at line 228 of file IpQualityFunctionMuOracle.hpp.

6.136.5.22 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_z_L_ [private]

Definition at line 229 of file IpQualityFunctionMuOracle.hpp.

6.136.5.23 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_z_U_ [private]

Definition at line 230 of file IpQualityFunctionMuOracle.hpp.

6.136.5.24 **SmartPtr<Vector>** Ipopt::QualityFunctionMuOracle::tmp_v_L_ [private]

Definition at line 231 of file IpQualityFunctionMuOracle.hpp.

6.136.5.25 `SmartPointer<Vector> Ipopt::QualityFunctionMuOracle::tmp_v_U_` [private]

Definition at line 232 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.26 `Index Ipopt::QualityFunctionMuOracle::count_qf_evals_` [private]

Definition at line 236 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.27 `bool Ipopt::QualityFunctionMuOracle::initialized_` [private]

Definition at line 243 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.28 `Index Ipopt::QualityFunctionMuOracle::n_dual_` [private]

Definition at line 244 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.29 `Index Ipopt::QualityFunctionMuOracle::n_pri_` [private]

Definition at line 245 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.30 `Index Ipopt::QualityFunctionMuOracle::n_comp_` [private]

Definition at line 246 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.31 `SmartPointer<const Vector> Ipopt::QualityFunctionMuOracle::curr_slack_x_L_` [private]

Definition at line 248 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.32 `SmartPointer<const Vector> Ipopt::QualityFunctionMuOracle::curr_slack_x_U_` [private]

Definition at line 249 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.33 `SmartPointer<const Vector> Ipopt::QualityFunctionMuOracle::curr_slack_s_L_` [private]

Definition at line 250 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.34 `SmartPointer<const Vector> Ipopt::QualityFunctionMuOracle::curr_slack_s_U_` [private]

Definition at line 251 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.35 `SmartPointer<const Vector> Ipopt::QualityFunctionMuOracle::curr_z_L_` [private]

Definition at line 253 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.36 `SmartPointer<const Vector> Ipopt::QualityFunctionMuOracle::curr_z_U_` [private]

Definition at line 254 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.37 `SmartPointer<const Vector> Ipopt::QualityFunctionMuOracle::curr_v_L_` [private]

Definition at line 255 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.38 `SmartPointer<const Vector> Ipopt::QualityFunctionMuOracle::curr_v_U_` [private]

Definition at line 256 of file `IpQualityFunctionMuOracle.hpp`.

6.136.5.39 **Number** Ipopt::QualityFunctionMuOracle::curr_grad_lag_x_asum_ [private]

Definition at line 258 of file IpQualityFunctionMuOracle.hpp.

6.136.5.40 **Number** Ipopt::QualityFunctionMuOracle::curr_grad_lag_s_asum_ [private]

Definition at line 259 of file IpQualityFunctionMuOracle.hpp.

6.136.5.41 **Number** Ipopt::QualityFunctionMuOracle::curr_c_asum_ [private]

Definition at line 260 of file IpQualityFunctionMuOracle.hpp.

6.136.5.42 **Number** Ipopt::QualityFunctionMuOracle::curr_d_minus_s_asum_ [private]

Definition at line 261 of file IpQualityFunctionMuOracle.hpp.

6.136.5.43 **Number** Ipopt::QualityFunctionMuOracle::curr_grad_lag_x_nrm2_ [private]

Definition at line 263 of file IpQualityFunctionMuOracle.hpp.

6.136.5.44 **Number** Ipopt::QualityFunctionMuOracle::curr_grad_lag_s_nrm2_ [private]

Definition at line 264 of file IpQualityFunctionMuOracle.hpp.

6.136.5.45 **Number** Ipopt::QualityFunctionMuOracle::curr_c_nrm2_ [private]

Definition at line 265 of file IpQualityFunctionMuOracle.hpp.

6.136.5.46 **Number** Ipopt::QualityFunctionMuOracle::curr_d_minus_s_nrm2_ [private]

Definition at line 266 of file IpQualityFunctionMuOracle.hpp.

6.136.5.47 **Number** Ipopt::QualityFunctionMuOracle::curr_grad_lag_x_amax_ [private]

Definition at line 268 of file IpQualityFunctionMuOracle.hpp.

6.136.5.48 **Number** Ipopt::QualityFunctionMuOracle::curr_grad_lag_s_amax_ [private]

Definition at line 269 of file IpQualityFunctionMuOracle.hpp.

6.136.5.49 **Number** Ipopt::QualityFunctionMuOracle::curr_c_amax_ [private]

Definition at line 270 of file IpQualityFunctionMuOracle.hpp.

6.136.5.50 **Number** Ipopt::QualityFunctionMuOracle::curr_d_minus_s_amax_ [private]

Definition at line 271 of file IpQualityFunctionMuOracle.hpp.

The documentation for this class was generated from the following file:

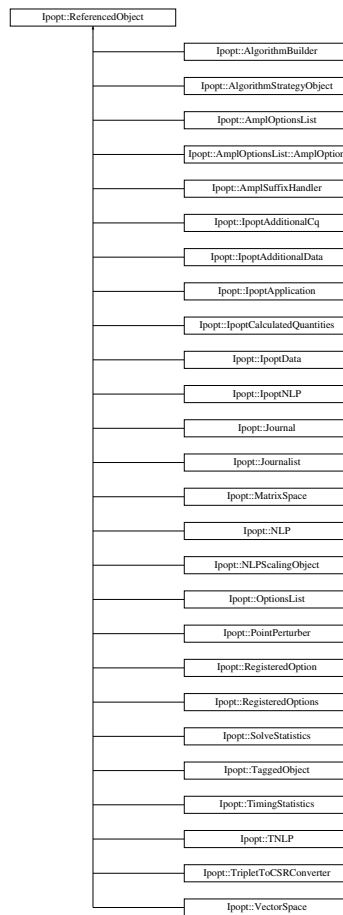
- [Algorithm/IpQualityFunctionMuOracle.hpp](#)

6.137 Ipopt::ReferencedObject Class Reference

[ReferencedObject](#) class.

```
#include <IpReferenced.hpp>
```

Inheritance diagram for `Ipopt::ReferencedObject`:



Public Member Functions

- [ReferencedObject](#) ()
- virtual [~ReferencedObject](#) ()
- [Index ReferenceCount](#) () const
- void [AddRef](#) (const [Referencer](#) *referencer) const
- void [ReleaseRef](#) (const [Referencer](#) *referencer) const

Private Attributes

- [Index reference_count_](#)

6.137.1 Detailed Description

[ReferencedObject](#) class.

This is part of the implementation of an intrusive smart pointer design. This class stores the reference count of all the smart pointers that currently reference it. See the documentation for the [SmartPtr](#) class for more details.

A [SmartPtr](#) behaves much like a raw pointer, but manages the lifetime of an object, deleting the object automatically. This class implements a reference-counting, intrusive smart pointer design, where all objects pointed to must inherit off

of [ReferencedObject](#), which stores the reference count. Although this is intrusive (native types and externally authored classes require wrappers to be referenced by smart pointers), it is a safer design. A more detailed discussion of these issues follows after the usage information.

Usage Example: Note: to use the [SmartPtr](#), all objects to which you point MUST inherit off of [ReferencedObject](#).

```
*
* In MyClass.hpp...
*
* #include "IpReferenced.hpp"
*
* namespace Ipopt {
*
*   class MyClass : public ReferencedObject // must derive from ReferencedObject
*   {
*     ...
*   }
* } // namespace Ipopt
*
*
* In my_usage.cpp...
*
* #include "IpSmartPtr.hpp"
* #include "MyClass.hpp"
*
* void func(AnyObject& obj)
* {
*   SmartPtr<MyClass> ptr_to_myclass = new MyClass(...);
*   // ptr_to_myclass now points to a new MyClass,
*   // and the reference count is 1
*
*   ...
*
*   obj.SetMyClass(ptr_to_myclass);
*   // Here, let's assume that AnyObject uses a
*   // SmartPtr<MyClass> internally here.
*   // Now, both ptr_to_myclass and the internal
*   // SmartPtr in obj point to the same MyClass object
*   // and its reference count is 2.
*
*   ...
*
*   // No need to delete ptr_to_myclass, this
*   // will be done automatically when the
*   // reference count drops to zero.
*
* }
```

Other Notes: The [SmartPtr](#) implements both dereference operators `->` & `*`. The [SmartPtr](#) does NOT implement a conversion operator to the raw pointer. Use the [GetRawPtr\(\)](#) method when this is necessary. Make sure that the raw pointer is NOT deleted. The [SmartPtr](#) implements the comparison operators `==` & `!=` for a variety of types. Use these instead of

```
*   if (GetRawPtr(smrt_ptr) == ptr) // Don't use this
*
```

[SmartPtr](#)'s, as currently implemented, do NOT handle circular references. For example: consider a higher level object using [SmartPtr](#)s to point to A and B, but A and B also point to each other (i.e. A has a [SmartPtr](#) to B and B has a [SmartPtr](#) to A). In this scenario, when the higher level object is finished with A and B, their reference counts will never drop to zero (since they reference each other) and they will not be deleted. This can be detected by memory leak tools like valgrind. If the circular reference is necessary, the problem can be overcome by a number of techniques:

- 1) A and B can have a method that "releases" each other, that is they set their internal [SmartPtr](#)s to NULL.

```

*         void AClass::ReleaseCircularReferences()
*         {
*             smart_ptr_to_B = NULL;
*         }
*

```

Then, the higher level class can call these methods before it is done using A & B.

2) Raw pointers can be used in A and B to reference each other. Here, an implicit assumption is made that the lifetime is controlled by the higher level object and that A and B will both exist in a controlled manner. Although this seems dangerous, in many situations, this type of referencing is very controlled and this is reasonably safe.

3) This [SmartPtr](#) class could be redesigned with the Weak/Strong design concept. Here, the [SmartPtr](#) is identified as being Strong (controls lifetime of the object) or Weak (merely referencing the object). The Strong [SmartPtr](#) increments (and decrements) the reference count in [ReferencedObject](#) but the Weak [SmartPtr](#) does not. In the example above, the higher level object would have Strong SmartPtrs to A and B, but A and B would have Weak SmartPtrs to each other. Then, when the higher level object was done with A and B, they would be deleted. The Weak SmartPtrs in A and B would not decrement the reference count and would, of course, not delete the object. This idea is very similar to item (2), where it is implied that the sequence of events is controlled such that A and B will not call anything using their pointers following the higher level delete (i.e. in their destructors!). This is somehow safer, however, because code can be written (however expensive) to perform run-time detection of this situation. For example, the [ReferencedObject](#) could store pointers to all Weak SmartPtrs that are referencing it and, in its destructor, tell these pointers that it is dying. They could then set themselves to NULL, or set an internal flag to detect usage past this point.

For every most derived object only one [ReferencedObject](#) may exist, that is multiple inheritance requires virtual inheritance, see also the 2nd point in ticket #162.

Comments on Non-Intrusive Design: In a non-intrusive design, the reference count is stored somewhere other than the object being referenced. This means, unless the reference counting pointer is the first referencer, it must get a pointer to the referenced object from another smart pointer (so it has access to the reference count location). In this non-intrusive design, if we are pointing to an object with a smart pointer (or a number of smart pointers), and we then give another smart pointer the address through a RAW pointer, we will have two independent, AND INCORRECT, reference counts. To avoid this pitfall, we use an intrusive reference counting technique where the reference count is stored in the object being referenced.

Definition at line 174 of file IpReferenced.hpp.

6.137.2 Constructor & Destructor Documentation

6.137.2.1 Ipopt::ReferencedObject::ReferencedObject () [inline]

Definition at line 177 of file IpReferenced.hpp.

6.137.2.2 virtual Ipopt::ReferencedObject::~~ReferencedObject () [inline],[virtual]

Definition at line 182 of file IpReferenced.hpp.

6.137.3 Member Function Documentation

6.137.3.1 Index Ipopt::ReferencedObject::ReferenceCount () const [inline]

Definition at line 207 of file IpReferenced.hpp.

6.137.3.2 void Ipopt::ReferencedObject::AddRef (const Referencer * referencer) const [inline]

Definition at line 215 of file IpReferenced.hpp.

6.137.3.3 `void Ipopt::ReferencedObject::ReleaseRef (const Referencer * referencer) const` `[inline]`

Definition at line 227 of file IpReferenced.hpp.

6.137.4 Member Data Documentation

6.137.4.1 `Index Ipopt::ReferencedObject::reference_count_` `[mutable],[private]`

Definition at line 197 of file IpReferenced.hpp.

The documentation for this class was generated from the following file:

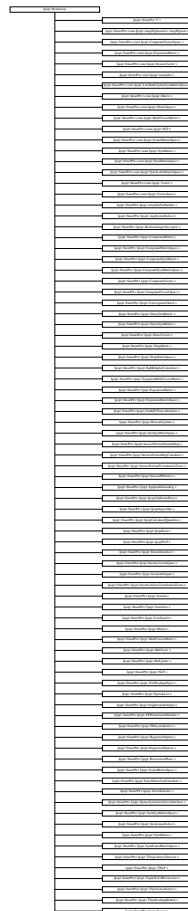
- [Common/IpReferenced.hpp](#)

6.138 Ipopt::Referencer Class Reference

Psydo-class, from which everything has to inherit that wants to use be registered as a [Referencer](#) for a [Referenced-Object](#).

```
#include <IpReferenced.hpp>
```

Inheritance diagram for Ipopt::Referencer:



6.138.1 Detailed Description

Psydo-class, from which everything has to inherit that wants to use be registered as a [Referencer](#) for a [Referenced-Object](#).

Definition at line 27 of file IpReferenced.hpp.

The documentation for this class was generated from the following file:

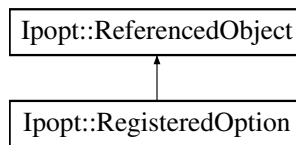
- [Common/IpReferenced.hpp](#)

6.139 Ipopt::RegisteredOption Class Reference

Base class for registered options.

```
#include <IpRegOptions.hpp>
```

Inheritance diagram for Ipopt::RegisteredOption:



Classes

- class [string_entry](#)
class to hold the valid string settings for a string option

Public Member Functions

- [DECLARE_STD_EXCEPTION](#) (ERROR_CONVERTING_STRING_TO_ENUM)
- virtual void [OutputDescription](#) (const [Journalist](#) &jnlst) const
output a description of the option
- virtual void [OutputShortDescription](#) (const [Journalist](#) &jnlst) const
output a more concise version
- virtual void [OutputLatexDescription](#) (const [Journalist](#) &jnlst) const
output a latex version
- [RegisteredOption](#) ([Index](#) counter)
Constructors / Destructors.
- [RegisteredOption](#) (const std::string &name, const std::string &short_description, const std::string &long_description, const std::string ®istering_category, [Index](#) counter)
- [RegisteredOption](#) (const [RegisteredOption](#) ©)
- virtual [~RegisteredOption](#) ()
- virtual const std::string & [Name](#) () const
Standard Get / Set Methods.
- virtual void [SetName](#) (const std::string &name)

- Set the option's name (tag in the input file)
 - virtual const std::string & [ShortDescription](#) () const
Get the short description.
 - virtual const std::string & [LongDescription](#) () const
Get the long description.
 - virtual void [SetShortDescription](#) (const std::string &short_description)
Set the short description.
 - virtual void [SetLongDescription](#) (const std::string &long_description)
Set the long description.
 - virtual const std::string & [RegisteringCategory](#) () const
Get the registering class.
 - virtual void [SetRegisteringCategory](#) (const std::string ®istering_category)
Set the registering class.
 - virtual const
[RegisteredOptionType](#) & [Type](#) () const
Get the Option's type.
 - virtual void [SetType](#) (const [RegisteredOptionType](#) &type)
Get the Option's type.
 - virtual [Index Counter](#) () const
Counter.

Get / Set methods valid for specific types - NOTE: the Type

must be set before calling these methods.

- virtual const bool & [HasLower](#) () const
check if the option has a lower bound - can be called for OT_Number & OT_Integer
- virtual const bool & [LowerStrict](#) () const
check if the lower bound is strict - can be called for OT_Number
- virtual [Number LowerNumber](#) () const
get the Number version of the lower bound - can be called for OT_Number
- virtual void [SetLowerNumber](#) (const [Number](#) &lower, const bool &strict)
set the Number version of the lower bound - can be called for OT_Number
- virtual [Index LowerInteger](#) () const
get the Integer version of the lower bound can be called for OT_Integer
- virtual void [SetLowerInteger](#) (const [Index](#) &lower)
set the Integer version of the lower bound - can be called for OT_Integer
- virtual const bool & [HasUpper](#) () const
check if the option has an upper bound - can be called for OT_Number & OT_Integer
- virtual const bool & [UpperStrict](#) () const
check if the upper bound is strict - can be called for OT_Number
- virtual [Number UpperNumber](#) () const
get the Number version of the upper bound - can be called for OT_Number
- virtual void [SetUpperNumber](#) (const [Number](#) &upper, const bool &strict)
set the Number version of the upper bound - can be called for OT_Number
- virtual [Index UpperInteger](#) () const
get the Integer version of the upper bound - can be called for OT_Integer
- virtual void [SetUpperInteger](#) (const [Index](#) &upper)
set the Integer version of the upper bound - can be called for OT_Integer
- virtual void [AddValidStringSetting](#) (const std::string value, const std::string description)
method to add valid string entries - can be called for OT_String
- virtual [Number DefaultNumber](#) () const

- get the default as a Number - can be called for OT_Number*
- virtual void [SetDefaultNumber](#) (const [Number](#) &default_value)
Set the default as a Number - can be called for OT_Number.
- virtual [Index DefaultInteger](#) () const
get the default as an Integer - can be called for OT_Integer
- virtual void [SetDefaultInteger](#) (const [Index](#) &default_value)
Set the default as an Integer - can be called for OT_Integer
- virtual std::string [DefaultString](#) () const
get the default as a string - can be called for OT_String
- virtual [Index DefaultStringAsEnum](#) () const
get the default as a string, but as the index of the string in the list - helps map from a string to an enum- can be called for OT_String
- virtual void [SetDefaultString](#) (const std::string &default_value)
Set the default as a string - can be called for OT_String.
- virtual std::vector< [string_entry](#) > [GetValidStrings](#) () const
get the valid string settings - can be called for OT_String
- virtual bool [IsValidNumberSetting](#) (const [Number](#) &value) const
Check if the Number value is a valid setting - can be called for OT_Number.
- virtual bool [IsValidIntegerSetting](#) (const [Index](#) &value) const
Check if the Integer value is a valid setting - can be called for OT_Integer.
- virtual bool [IsValidStringSetting](#) (const std::string &value) const
Check if the String value is a valid setting - can be called for OT_String.
- virtual std::string [MapStringSetting](#) (const std::string &value) const
Map a user setting (allowing any case) to the case used when the setting was registered.
- virtual [Index MapStringSettingToEnum](#) (const std::string &value) const
Map a user setting (allowing any case) to the index of the matched setting in the list of string settings.

Private Member Functions

- void [MakeValidLatexString](#) (std::string source, std::string &dest) const
- std::string [MakeValidLatexNumber](#) ([Number](#) value) const
- bool [string_equal_insensitive](#) (const std::string &s1, const std::string &s2) const
Compare two strings and return true if they are equal (case insensitive comparison)

Private Attributes

- std::string [name_](#)
- std::string [short_description_](#)
- std::string [long_description_](#)
- std::string [registering_category_](#)
- [RegisteredOptionType](#) [type_](#)
- bool [has_lower_](#)
- bool [lower_strict_](#)
- [Number](#) [lower_](#)
- bool [has_upper_](#)
- bool [upper_strict_](#)
- [Number](#) [upper_](#)
- [Number](#) [default_number_](#)
- std::vector< [string_entry](#) > [valid_strings_](#)
- std::string [default_string_](#)
- const [Index](#) [counter_](#)

Has the information as how many-th option this one was registered.

6.139.1 Detailed Description

Base class for registered options.

The derived types are more specific to a string option or a Number (real) option, etc.

Definition at line 33 of file IpRegOptions.hpp.

6.139.2 Constructor & Destructor Documentation

6.139.2.1 Ipopt::RegisteredOption::RegisteredOption (Index counter) [inline]

Constructors / Destructors.

Definition at line 49 of file IpRegOptions.hpp.

6.139.2.2 Ipopt::RegisteredOption::RegisteredOption (const std::string & name, const std::string & short_description, const std::string & long_description, const std::string & registering_category, Index counter) [inline]

Definition at line 57 of file IpRegOptions.hpp.

6.139.2.3 Ipopt::RegisteredOption::RegisteredOption (const RegisteredOption & copy) [inline]

Definition at line 73 of file IpRegOptions.hpp.

6.139.2.4 virtual Ipopt::RegisteredOption::~~RegisteredOption () [inline],[virtual]

Definition at line 88 of file IpRegOptions.hpp.

6.139.3 Member Function Documentation

6.139.3.1 Ipopt::RegisteredOption::DECLARE_STD_EXCEPTION (ERROR_CONVERTING_STRING_TO_ENUM)

6.139.3.2 virtual const std::string& Ipopt::RegisteredOption::Name () const [inline],[virtual]

Standard Get / Set Methods.

Get the option's name (tag in the input file)

Definition at line 97 of file IpRegOptions.hpp.

6.139.3.3 virtual void Ipopt::RegisteredOption::SetName (const std::string & name) [inline],[virtual]

Set the option's name (tag in the input file)

Definition at line 102 of file IpRegOptions.hpp.

6.139.3.4 virtual const std::string& Ipopt::RegisteredOption::ShortDescription () const [inline],[virtual]

Get the short description.

Definition at line 107 of file IpRegOptions.hpp.

6.139.3.5 virtual const std::string& Ipopt::RegisteredOption::LongDescription () const [inline],[virtual]

Get the long description.

Definition at line 112 of file IpRegOptions.hpp.

6.139.3.6 `virtual void Ipopt::RegisteredOption::SetShortDescription (const std::string & short_description) [inline],
[virtual]`

Set the short description.

Definition at line 117 of file IpRegOptions.hpp.

6.139.3.7 `virtual void Ipopt::RegisteredOption::SetLongDescription (const std::string & long_description) [inline],
[virtual]`

Set the long description.

Definition at line 122 of file IpRegOptions.hpp.

6.139.3.8 `virtual const std::string& Ipopt::RegisteredOption::RegisteringCategory () const [inline],[virtual]`

Get the registering class.

Definition at line 127 of file IpRegOptions.hpp.

6.139.3.9 `virtual void Ipopt::RegisteredOption::SetRegisteringCategory (const std::string & registering_category) [inline],
[virtual]`

Set the registering class.

Definition at line 132 of file IpRegOptions.hpp.

6.139.3.10 `virtual const RegisteredOptionType& Ipopt::RegisteredOption::Type () const [inline],[virtual]`

Get the Option's type.

Definition at line 137 of file IpRegOptions.hpp.

6.139.3.11 `virtual void Ipopt::RegisteredOption::SetType (const RegisteredOptionType & type) [inline],
[virtual]`

Get the Option's type.

Definition at line 142 of file IpRegOptions.hpp.

6.139.3.12 `virtual Index Ipopt::RegisteredOption::Counter () const [inline],[virtual]`

Counter.

Definition at line 147 of file IpRegOptions.hpp.

6.139.3.13 `virtual const bool& Ipopt::RegisteredOption::HasLower () const [inline],[virtual]`

check if the option has a lower bound - can be called for OT_Number & OT_Integer

Definition at line 159 of file IpRegOptions.hpp.

6.139.3.14 `virtual const bool& Ipopt::RegisteredOption::LowerStrict () const [inline],[virtual]`

check if the lower bound is strict - can be called for

OT_Number

Definition at line 166 of file IpRegOptions.hpp.

6.139.3.15 `virtual Number Ipopt::RegisteredOption::LowerNumber () const [inline],[virtual]`

get the Number version of the lower bound - can be called for OT_Number

Definition at line 173 of file IpRegOptions.hpp.

6.139.3.16 `virtual void Ipopt::RegisteredOption::SetLowerNumber (const Number & lower, const bool & strict) [inline],[virtual]`

set the Number version of the lower bound - can be called for OT_Number

Definition at line 180 of file IpRegOptions.hpp.

6.139.3.17 `virtual Index Ipopt::RegisteredOption::LowerInteger () const [inline],[virtual]`

get the Integer version of the lower bound can be called for OT_Integer

Definition at line 188 of file IpRegOptions.hpp.

6.139.3.18 `virtual void Ipopt::RegisteredOption::SetLowerInteger (const Index & lower) [inline],[virtual]`

set the Integer version of the lower bound - can be called for OT_Integer

Definition at line 195 of file IpRegOptions.hpp.

6.139.3.19 `virtual const bool& Ipopt::RegisteredOption::HasUpper () const [inline],[virtual]`

check if the option has an upper bound - can be called for OT_Number & OT_Integer

Definition at line 203 of file IpRegOptions.hpp.

6.139.3.20 `virtual const bool& Ipopt::RegisteredOption::UpperStrict () const [inline],[virtual]`

check if the upper bound is strict - can be called for OT_Number

Definition at line 210 of file IpRegOptions.hpp.

6.139.3.21 `virtual Number Ipopt::RegisteredOption::UpperNumber () const [inline],[virtual]`

get the Number version of the upper bound - can be called for OT_Number

Definition at line 217 of file IpRegOptions.hpp.

6.139.3.22 `virtual void Ipopt::RegisteredOption::SetUpperNumber (const Number & upper, const bool & strict) [inline],[virtual]`

set the Number version of the upper bound - can be called for OT_Number

Definition at line 224 of file IpRegOptions.hpp.

6.139.3.23 `virtual Index Ipopt::RegisteredOption::UpperInteger () const [inline],[virtual]`

get the Integer version of the upper bound - can be called for OT_Integer

Definition at line 233 of file IpRegOptions.hpp.

6.139.3.24 `virtual void Ipopt::RegisteredOption::SetUpperInteger (const Index & upper) [inline],[virtual]`

set the Integer version of the upper bound - can be called for OT_Integer

Definition at line 240 of file IpRegOptions.hpp.

6.139.3.25 `virtual void Ipopt::RegisteredOption::AddValidStringSetting (const std::string value, const std::string description)`
`[inline], [virtual]`

method to add valid string entries - can be called for OT_String

Definition at line 248 of file IpRegOptions.hpp.

6.139.3.26 `virtual Number Ipopt::RegisteredOption::DefaultNumber () const` `[inline], [virtual]`

get the default as a Number - can be called for OT_Number

Definition at line 255 of file IpRegOptions.hpp.

6.139.3.27 `virtual void Ipopt::RegisteredOption::SetDefaultNumber (const Number & default_value)` `[inline],`
`[virtual]`

Set the default as a Number - can be called for OT_Number.

Definition at line 261 of file IpRegOptions.hpp.

6.139.3.28 `virtual Index Ipopt::RegisteredOption::DefaultInteger () const` `[inline], [virtual]`

get the default as an Integer - can be called for OT_Integer

Definition at line 267 of file IpRegOptions.hpp.

6.139.3.29 `virtual void Ipopt::RegisteredOption::SetDefaultInteger (const Index & default_value)` `[inline], [virtual]`

Set the default as an Integer - can be called for

OT_Integer

Definition at line 274 of file IpRegOptions.hpp.

6.139.3.30 `virtual std::string Ipopt::RegisteredOption::DefaultString () const` `[inline], [virtual]`

get the default as a string - can be called for OT_String

Definition at line 280 of file IpRegOptions.hpp.

6.139.3.31 `virtual Index Ipopt::RegisteredOption::DefaultStringAsEnum () const` `[inline], [virtual]`

get the default as a string, but as the index of the string in the list - helps map from a string to an enum- can be called for OT_String

Definition at line 288 of file IpRegOptions.hpp.

6.139.3.32 `virtual void Ipopt::RegisteredOption::SetDefaultString (const std::string & default_value)` `[inline],`
`[virtual]`

Set the default as a string - can be called for OT_String.

Definition at line 294 of file IpRegOptions.hpp.

6.139.3.33 `virtual std::vector<string_entry> Ipopt::RegisteredOption::GetValidStrings () const` `[inline], [virtual]`

get the valid string settings - can be called for OT_String

Definition at line 300 of file IpRegOptions.hpp.

6.139.3.34 `virtual bool Ipopt::RegisteredOption::IsValidNumberSetting (const Number & value) const [inline],
[virtual]`

Check if the Number value is a valid setting - can be called for OT_Number.

Definition at line 307 of file IpRegOptions.hpp.

6.139.3.35 `virtual bool Ipopt::RegisteredOption::IsValidIntegerSetting (const Index & value) const [inline], [virtual]`

Check if the Integer value is a valid setting - can be called for OT_Integer.

Definition at line 322 of file IpRegOptions.hpp.

6.139.3.36 `virtual bool Ipopt::RegisteredOption::IsValidStringSetting (const std::string & value) const [virtual]`

Check if the String value is a valid setting - can be called for OT_String.

6.139.3.37 `virtual std::string Ipopt::RegisteredOption::MapStringSetting (const std::string & value) const [virtual]`

Map a user setting (allowing any case) to the case used when the setting was registered.

6.139.3.38 `virtual Index Ipopt::RegisteredOption::MapStringSettingToEnum (const std::string & value) const [virtual]`

Map a user setting (allowing any case) to the index of the matched setting in the list of string settings.

Helps map a string setting to an enumeration.

6.139.3.39 `virtual void Ipopt::RegisteredOption::OutputDescription (const Journalist & jnlst) const [virtual]`

output a description of the option

6.139.3.40 `virtual void Ipopt::RegisteredOption::OutputShortDescription (const Journalist & jnlst) const [virtual]`

output a more concise version

6.139.3.41 `virtual void Ipopt::RegisteredOption::OutputLatexDescription (const Journalist & jnlst) const [virtual]`

output a latex version

6.139.3.42 `void Ipopt::RegisteredOption::MakeValidLatexString (std::string source, std::string & dest) const [private]`

6.139.3.43 `std::string Ipopt::RegisteredOption::MakeValidLatexNumber (Number value) const [private]`

6.139.3.44 `bool Ipopt::RegisteredOption::string_equal_insensitive (const std::string & s1, const std::string & s2) const
[private]`

Compare two strings and return true if they are equal (case

insensitive comparison)

6.139.4 Member Data Documentation

6.139.4.1 `std::string Ipopt::RegisteredOption::name_ [private]`

Definition at line 357 of file IpRegOptions.hpp.

6.139.4.2 `std::string Ipopt::RegisteredOption::short_description_` [private]

Definition at line 358 of file IpRegOptions.hpp.

6.139.4.3 `std::string Ipopt::RegisteredOption::long_description_` [private]

Definition at line 359 of file IpRegOptions.hpp.

6.139.4.4 `std::string Ipopt::RegisteredOption::registering_category_` [private]

Definition at line 360 of file IpRegOptions.hpp.

6.139.4.5 `RegisteredOptionType Ipopt::RegisteredOption::type_` [private]

Definition at line 361 of file IpRegOptions.hpp.

6.139.4.6 `bool Ipopt::RegisteredOption::has_lower_` [private]

Definition at line 363 of file IpRegOptions.hpp.

6.139.4.7 `bool Ipopt::RegisteredOption::lower_strict_` [private]

Definition at line 364 of file IpRegOptions.hpp.

6.139.4.8 `Number Ipopt::RegisteredOption::lower_` [private]

Definition at line 365 of file IpRegOptions.hpp.

6.139.4.9 `bool Ipopt::RegisteredOption::has_upper_` [private]

Definition at line 366 of file IpRegOptions.hpp.

6.139.4.10 `bool Ipopt::RegisteredOption::upper_strict_` [private]

Definition at line 367 of file IpRegOptions.hpp.

6.139.4.11 `Number Ipopt::RegisteredOption::upper_` [private]

Definition at line 368 of file IpRegOptions.hpp.

6.139.4.12 `Number Ipopt::RegisteredOption::default_number_` [private]

Definition at line 369 of file IpRegOptions.hpp.

6.139.4.13 `std::vector<string_entry> Ipopt::RegisteredOption::valid_strings_` [private]

Definition at line 379 of file IpRegOptions.hpp.

6.139.4.14 `std::string Ipopt::RegisteredOption::default_string_` [private]

Definition at line 380 of file IpRegOptions.hpp.

6.139.4.15 `const Index Ipopt::RegisteredOption::counter_` [private]

Has the information as how many-th option this one was registered.

Definition at line 384 of file IpRegOptions.hpp.

The documentation for this class was generated from the following file:

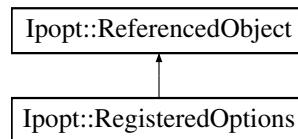
- [Common/IpRegOptions.hpp](#)

6.140 Ipopt::RegisteredOptions Class Reference

Class for storing registered options.

```
#include <IpRegOptions.hpp>
```

Inheritance diagram for Ipopt::RegisteredOptions:



Public Types

- `typedef std::map< std::string, SmartPtr< RegisteredOption > > RegOptionsList`

Public Member Functions

- [DECLARE_STD_EXCEPTION](#) (OPTION_ALREADY_REGISTERED)
- virtual const [RegOptionsList](#) & [RegisteredOptionsList](#) () const
Giving access to iterable representation of the registered options.
- [RegisteredOptions](#) ()
Constructors / Destructors.
- virtual [~RegisteredOptions](#) ()
Standard Destructor.
- virtual void [SetRegisteringCategory](#) (const std::string ®istering_category)
Methods to interact with registered options.
- virtual std::string [RegisteringCategory](#) ()
retrieve the value of the current registering category
- virtual void [AddNumberOption](#) (const std::string &name, const std::string &short_description, [Number](#) default_value, const std::string &long_description="")
Add a Number option (with no restrictions)
- virtual void [AddLowerBoundedNumberOption](#) (const std::string &name, const std::string &short_description, [Number](#) lower, bool strict, [Number](#) default_value, const std::string &long_description="")
Add a Number option (with a lower bound)
- virtual void [AddUpperBoundedNumberOption](#) (const std::string &name, const std::string &short_description, [Number](#) upper, bool strict, [Number](#) default_value, const std::string &long_description="")
Add a Number option (with a upper bound)
- virtual void [AddBoundedNumberOption](#) (const std::string &name, const std::string &short_description, [Number](#) lower, bool lower_strict, [Number](#) upper, bool upper_strict, [Number](#) default_value, const std::string &long_description="")
Add a Number option (with a both bounds)

- virtual void [AddStringOption9](#) (const std::string &name, const std::string &short_description, const std::string &default_value, const std::string &setting1, const std::string &description1, const std::string &setting2, const std::string &description2, const std::string &setting3, const std::string &description3, const std::string &setting4, const std::string &description4, const std::string &setting5, const std::string &description5, const std::string &setting6, const std::string &description6, const std::string &setting7, const std::string &description7, const std::string &setting8, const std::string &description8, const std::string &setting9, const std::string &description9, const std::string &long_description="")
- virtual void [AddStringOption10](#) (const std::string &name, const std::string &short_description, const std::string &default_value, const std::string &setting1, const std::string &description1, const std::string &setting2, const std::string &description2, const std::string &setting3, const std::string &description3, const std::string &setting4, const std::string &description4, const std::string &setting5, const std::string &description5, const std::string &setting6, const std::string &description6, const std::string &setting7, const std::string &description7, const std::string &setting8, const std::string &description8, const std::string &setting9, const std::string &description9, const std::string &setting10, const std::string &description10, const std::string &long_description="")
- virtual [SmartPtr](#)< const [RegisteredOption](#) > [GetOption](#) (const std::string &name)
Get a registered option - this will return NULL if the option does not exist.
- virtual void [OutputOptionDocumentation](#) (const [Journalist](#) &jnlst, std::list< std::string > &categories)
Output documentation for the options - gives a description, etc.
- virtual void [OutputLatexOptionDocumentation](#) (const [Journalist](#) &jnlst, std::list< std::string > &categories)
Output documentation in Latex format to include in a latex file.

Private Attributes

- [Index](#) next_counter_
- std::string current_registering_category_
- std::map< std::string, [SmartPtr](#)< [RegisteredOption](#) > > registered_options_

6.140.1 Detailed Description

Class for storing registered options.

Used for validation and documentation.

Definition at line 390 of file IpRegOptions.hpp.

6.140.2 Member Typedef Documentation

6.140.2.1 `typedef std::map<std::string, SmartPtr<RegisteredOption> > Ipopt::RegisteredOptions::RegOptionsList`

Definition at line 642 of file IpRegOptions.hpp.

6.140.3 Constructor & Destructor Documentation

6.140.3.1 `Ipopt::RegisteredOptions::RegisteredOptions ()` `[inline]`

Constructors / Destructors.

Standard Constructor

Definition at line 396 of file IpRegOptions.hpp.

6.140.3.2 `virtual Ipopt::RegisteredOptions::~~RegisteredOptions () [inline],[virtual]`

Standard Destructor.

Definition at line 403 of file IpRegOptions.hpp.

6.140.4 Member Function Documentation

6.140.4.1 `Ipopt::RegisteredOptions::DECLARE_STD_EXCEPTION (OPTION_ALREADY_REGISTERED)`

6.140.4.2 `virtual void Ipopt::RegisteredOptions::SetRegisteringCategory (const std::string & registering_category) [inline],[virtual]`

Methods to interact with registered options.

set the registering class. All subsequent options will be added with the registered class

Definition at line 413 of file IpRegOptions.hpp.

6.140.4.3 `virtual std::string Ipopt::RegisteredOptions::RegisteringCategory () [inline],[virtual]`

retrieve the value of the current registering category

Definition at line 419 of file IpRegOptions.hpp.

6.140.4.4 `virtual void Ipopt::RegisteredOptions::AddNumberOption (const std::string & name, const std::string & short_description, Number default_value, const std::string & long_description = " ") [virtual]`

Add a Number option (with no restrictions)

6.140.4.5 `virtual void Ipopt::RegisteredOptions::AddLowerBoundedNumberOption (const std::string & name, const std::string & short_description, Number lower, bool strict, Number default_value, const std::string & long_description = " ") [virtual]`

Add a Number option (with a lower bound)

6.140.4.6 `virtual void Ipopt::RegisteredOptions::AddUpperBoundedNumberOption (const std::string & name, const std::string & short_description, Number upper, bool strict, Number default_value, const std::string & long_description = " ") [virtual]`

Add a Number option (with a upper bound)

6.140.4.7 `virtual void Ipopt::RegisteredOptions::AddBoundedNumberOption (const std::string & name, const std::string & short_description, Number lower, bool lower_strict, Number upper, bool upper_strict, Number default_value, const std::string & long_description = " ") [virtual]`

Add a Number option (with a both bounds)

6.140.4.8 `virtual void Ipopt::RegisteredOptions::AddIntegerOption (const std::string & name, const std::string & short_description, Index default_value, const std::string & long_description = " ") [virtual]`

Add a Integer option (with no restrictions)

6.140.4.9 `virtual void Ipopt::RegisteredOptions::AddLowerBoundedIntegerOption (const std::string & name, const std::string & short_description, Index lower, Index default_value, const std::string & long_description = " ") [virtual]`

Add a Integer option (with a lower bound)

6.140.4.10 `virtual void Ipopt::RegisteredOptions::AddUpperBoundedIntegerOption (const std::string & name, const std::string & short_description, Index upper, Index default_value, const std::string & long_description = " ") [virtual]`

Add a Integer option (with a upper bound)

6.140.4.11 `virtual void Ipopt::RegisteredOptions::AddBoundedIntegerOption (const std::string & name, const std::string & short_description, Index lower, Index upper, Index default_value, const std::string & long_description = " ") [virtual]`

Add a Integer option (with a both bounds)

6.140.4.12 `virtual void Ipopt::RegisteredOptions::AddStringOption (const std::string & name, const std::string & short_description, const std::string & default_value, const std::vector< std::string > & settings, const std::vector< std::string > & descriptions, const std::string & long_description = " ") [virtual]`

Add a String option (with no restrictions)

6.140.4.13 `virtual void Ipopt::RegisteredOptions::AddStringOption1 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & long_description = " ") [virtual]`

Methods that make adding string options with only a few entries easier.

6.140.4.14 `virtual void Ipopt::RegisteredOptions::AddStringOption2 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & long_description = " ") [virtual]`

6.140.4.15 `virtual void Ipopt::RegisteredOptions::AddStringOption3 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & setting3, const std::string & description3, const std::string & long_description = " ") [virtual]`

6.140.4.16 `virtual void Ipopt::RegisteredOptions::AddStringOption4 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & setting3, const std::string & description3, const std::string & setting4, const std::string & description4, const std::string & long_description = " ") [virtual]`

6.140.4.17 `virtual void Ipopt::RegisteredOptions::AddStringOption5 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & setting3, const std::string & description3, const std::string & setting4, const std::string & description4, const std::string & setting5, const std::string & description5, const std::string & long_description = " ") [virtual]`

6.140.4.18 `virtual void Ipopt::RegisteredOptions::AddStringOption6 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & setting3, const std::string & description3, const std::string & setting4, const std::string & description4, const std::string & setting5, const std::string & description5, const std::string & setting6, const std::string & description6, const std::string & long_description = " ") [virtual]`

- 6.140.4.19 `virtual void Ipopt::RegisteredOptions::AddStringOption7 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & setting3, const std::string & description3, const std::string & setting4, const std::string & description4, const std::string & setting5, const std::string & description5, const std::string & setting6, const std::string & description6, const std::string & setting7, const std::string & description7, const std::string & long_description = " ") [virtual]`
- 6.140.4.20 `virtual void Ipopt::RegisteredOptions::AddStringOption8 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & setting3, const std::string & description3, const std::string & setting4, const std::string & description4, const std::string & setting5, const std::string & description5, const std::string & setting6, const std::string & description6, const std::string & setting7, const std::string & description7, const std::string & setting8, const std::string & description8, const std::string & long_description = " ") [virtual]`
- 6.140.4.21 `virtual void Ipopt::RegisteredOptions::AddStringOption9 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & setting3, const std::string & description3, const std::string & setting4, const std::string & description4, const std::string & setting5, const std::string & description5, const std::string & setting6, const std::string & description6, const std::string & setting7, const std::string & description7, const std::string & setting8, const std::string & description8, const std::string & setting9, const std::string & description9, const std::string & long_description = " ") [virtual]`
- 6.140.4.22 `virtual void Ipopt::RegisteredOptions::AddStringOption10 (const std::string & name, const std::string & short_description, const std::string & default_value, const std::string & setting1, const std::string & description1, const std::string & setting2, const std::string & description2, const std::string & setting3, const std::string & description3, const std::string & setting4, const std::string & description4, const std::string & setting5, const std::string & description5, const std::string & setting6, const std::string & description6, const std::string & setting7, const std::string & description7, const std::string & setting8, const std::string & description8, const std::string & setting9, const std::string & description9, const std::string & setting10, const std::string & description10, const std::string & long_description = " ") [virtual]`
- 6.140.4.23 `virtual SmartPtr<const RegisteredOption> Ipopt::RegisteredOptions::GetOption (const std::string & name) [virtual]`

Get a registered option - this will return NULL if the option does not exist.

- 6.140.4.24 `virtual void Ipopt::RegisteredOptions::OutputOptionDocumentation (const Journalist & jnlst, std::list< std::string > & categories) [virtual]`

Output documentation for the options - gives a description, etc.

- 6.140.4.25 `virtual void Ipopt::RegisteredOptions::OutputLatexOptionDocumentation (const Journalist & jnlst, std::list< std::string > & categories) [virtual]`

Output documentation in Latex format to include in a latex file.

- 6.140.4.26 `virtual const RegOptionsList& Ipopt::RegisteredOptions::RegisteredOptionsList () const [inline], [virtual]`

Giving access to iterable representation of the registered options.

Definition at line 646 of file IpRegOptions.hpp.

6.140.5 Member Data Documentation

6.140.5.1 Index Ipopt::RegisteredOptions::next_counter_ [private]

Definition at line 652 of file IpRegOptions.hpp.

6.140.5.2 std::string Ipopt::RegisteredOptions::current_registering_category_ [private]

Definition at line 653 of file IpRegOptions.hpp.

6.140.5.3 std::map<std::string, SmartPtr<RegisteredOption> > Ipopt::RegisteredOptions::registered_options_ [private]

Definition at line 654 of file IpRegOptions.hpp.

The documentation for this class was generated from the following file:

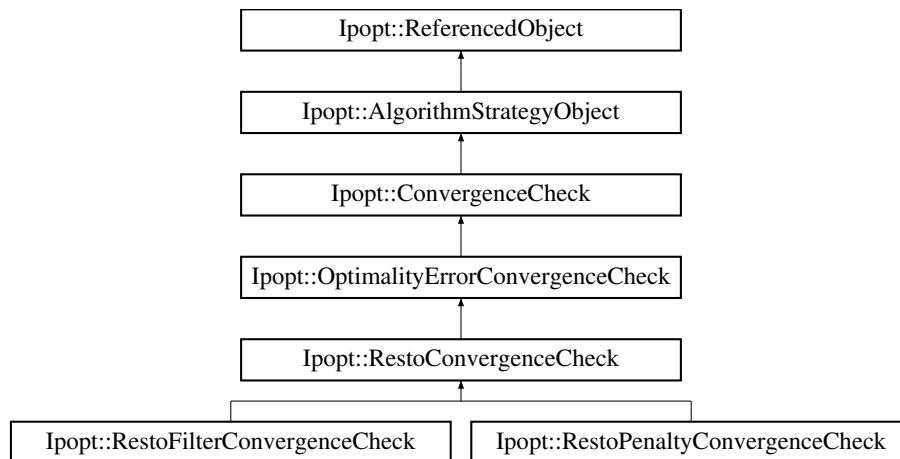
- Common/[IpRegOptions.hpp](#)

6.141 Ipopt::RestoConvergenceCheck Class Reference

Convergence check for the restoration phase.

```
#include <IpRestoConvCheck.hpp>
```

Inheritance diagram for Ipopt::RestoConvergenceCheck:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual [ConvergenceStatus](#) [CheckConvergence](#) (bool call_intermediate_callback=true)
overloaded from [ConvergenceCheck](#)
- virtual void [SetOrigLSAcceptor](#) (const [BacktrackingLSAcceptor](#) &orig_ls_acceptor)=0
Method for setting the LS acceptor from the main algorithm.

Constructors/Destructors

- [RestoConvergenceCheck](#) ()
Default Constructor.

- virtual [~RestoConvergenceCheck](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (SmartPtr< [RegisteredOptions](#) > roptions)
Methods used by IpoptType.

Private Member Functions

- virtual [ConvergenceStatus TestOrigProgress](#) (Number orig_trial_barr, Number orig_trial_theta)=0
Method for checking progress with original globalization mechanism.

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [RestoConvergenceCheck](#) (const [RestoConvergenceCheck](#) &)
Copy Constructor.
- void [operator=](#) (const [RestoConvergenceCheck](#) &)
Overloaded Equals Operator.

Private Attributes

- bool [first_resto_iter_](#)
Flag indicating that this is the first call.
- [Index successive_resto_iter_](#)
Counter for successive iterations in restoration phase.

Algorithmic parameters

- [Number kappa_resto_](#)
Fraction of required reduction in infeasibility before problem is considered to be solved.
- [Index maximum_iters_](#)
Maximum number of iterations in restoration phase.
- [Index maximum_resto_iters_](#)
Maximum number of successive iterations in restoration phase.
- [Number orig_constr_viol_tol_](#)
Constraint violation tolerance for original algorithm.

Additional Inherited Members

6.141.1 Detailed Description

Convergence check for the restoration phase.

This inherits from the [OptimalityErrorConvergenceCheck](#) so that the method for the regular optimality error convergence criterion can be checked as well. In addition, this convergence check returns the CONVERGED message, if the current iteration is acceptable to the original globalization scheme.

Definition at line 29 of file IpRestoConvCheck.hpp.

6.141.2 Constructor & Destructor Documentation

6.141.2.1 Ipopt::RestoConvergenceCheck::RestoConvergenceCheck ()

Default Constructor.

6.141.2.2 virtual Ipopt::RestoConvergenceCheck::~~RestoConvergenceCheck () [virtual]

Default destructor.

6.141.2.3 Ipopt::RestoConvergenceCheck::RestoConvergenceCheck (const RestoConvergenceCheck &) [private]

Copy Constructor.

6.141.3 Member Function Documentation

6.141.3.1 virtual bool Ipopt::RestoConvergenceCheck::InitializImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Reimplemented from [Ipopt::OptimalityErrorConvergenceCheck](#).

Reimplemented in [Ipopt::RestoFilterConvergenceCheck](#), and [Ipopt::RestoPenaltyConvergenceCheck](#).

6.141.3.2 virtual ConvergenceStatus Ipopt::RestoConvergenceCheck::CheckConvergence (bool call_intermediate_callback = true) [virtual]

overloaded from [ConvergenceCheck](#)

Reimplemented from [Ipopt::OptimalityErrorConvergenceCheck](#).

6.141.3.3 virtual void Ipopt::RestoConvergenceCheck::SetOrigLSAcceptor (const BacktrackingLSAcceptor & orig_ls_acceptor) [pure virtual]

Method for setting the LS acceptor from the main algorithm.

Implemented in [Ipopt::RestoFilterConvergenceCheck](#), and [Ipopt::RestoPenaltyConvergenceCheck](#).

6.141.3.4 static void Ipopt::RestoConvergenceCheck::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]

Methods used by IpoptType.

6.141.3.5 void Ipopt::RestoConvergenceCheck::operator= (const RestoConvergenceCheck &) [private]

Overloaded Equals Operator.

6.141.3.6 virtual ConvergenceStatus Ipopt::RestoConvergenceCheck::TestOrigProgress (Number orig_trial_barr, Number orig_trial_theta) [private],[pure virtual]

Method for checking progress with original globalization mechanism.

This needs to be overloaded

Implemented in [Ipopt::RestoFilterConvergenceCheck](#), and [Ipopt::RestoPenaltyConvergenceCheck](#).

6.141.4 Member Data Documentation

6.141.4.1 Number `Ipopt::RestoConvergenceCheck::kappa_resto_` [private]

Fraction of required reduction in infeasibility before problem is considered to be solved.

Definition at line 79 of file `IpRestoConvCheck.hpp`.

6.141.4.2 Index `Ipopt::RestoConvergenceCheck::maximum_iters_` [private]

Maximum number of iterations in restoration phase.

Definition at line 81 of file `IpRestoConvCheck.hpp`.

6.141.4.3 Index `Ipopt::RestoConvergenceCheck::maximum_resto_iters_` [private]

Maximum number of successive iterations in restoration phase.

Definition at line 83 of file `IpRestoConvCheck.hpp`.

6.141.4.4 Number `Ipopt::RestoConvergenceCheck::orig_constr_viol_tol_` [private]

Constraint violation tolerance for original algorithm.

Definition at line 85 of file `IpRestoConvCheck.hpp`.

6.141.4.5 bool `Ipopt::RestoConvergenceCheck::first_resto_iter_` [private]

Flag indicating that this is the first call.

We don't want to leave the restoration phase without taking at least one step, so this flag is used to ensure this.

Definition at line 91 of file `IpRestoConvCheck.hpp`.

6.141.4.6 Index `Ipopt::RestoConvergenceCheck::successive_resto_iter_` [private]

Counter for successive iterations in restoration phase.

Definition at line 94 of file `IpRestoConvCheck.hpp`.

The documentation for this class was generated from the following file:

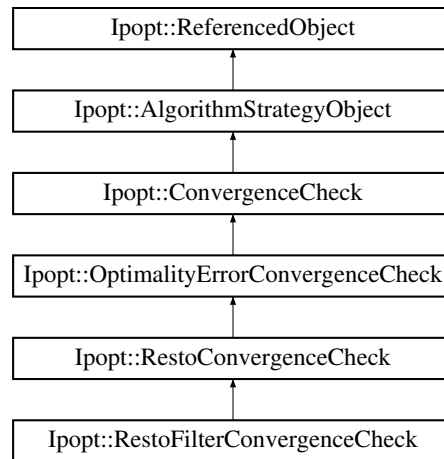
- [Algorithm/IpRestoConvCheck.hpp](#)

6.142 `Ipopt::RestoFilterConvergenceCheck` Class Reference

This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism.

```
#include <IpRestoFilterConvCheck.hpp>
```

Inheritance diagram for `Ipopt::RestoFilterConvergenceCheck`:



Public Member Functions

- void [SetOrigLSAcceptor](#) (const [BacktrackingLSAcceptor](#) &orig_ls_acceptor)
Set the object for the original filter line search.
- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Constructors/Destructors

- [RestoFilterConvergenceCheck](#) ()
Default Constructor.
- virtual [~RestoFilterConvergenceCheck](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods used by IpoptType.

Private Member Functions

- virtual [ConvergenceStatus](#) [TestOrigProgress](#) ([Number](#) orig_trial_barr, [Number](#) orig_trial_theta)
Method for checking progress with original filter globalization mechanism.

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [RestoFilterConvergenceCheck](#) (const [RestoFilterConvergenceCheck](#) &)
Copy Constructor.
- void [operator=](#) (const [RestoFilterConvergenceCheck](#) &)
Overloaded Equals Operator.

Private Attributes

- `const FilterLSAcceptor * orig_filter_ls_acceptor_`
Strategy object for the filter line search method for the original [NLP](#).

Additional Inherited Members

6.142.1 Detailed Description

This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism.

Definition at line 24 of file `IpRestoFilterConvCheck.hpp`.

6.142.2 Constructor & Destructor Documentation

6.142.2.1 `Ipopt::RestoFilterConvergenceCheck::RestoFilterConvergenceCheck ()`

Default Constructor.

6.142.2.2 `virtual Ipopt::RestoFilterConvergenceCheck::~~RestoFilterConvergenceCheck () [virtual]`

Default destructor.

6.142.2.3 `Ipopt::RestoFilterConvergenceCheck::RestoFilterConvergenceCheck (const RestoFilterConvergenceCheck &) [private]`

Copy Constructor.

6.142.3 Member Function Documentation

6.142.3.1 `void Ipopt::RestoFilterConvergenceCheck::SetOrigLSAcceptor (const BacktrackingLSAcceptor & orig_ls_acceptor) [virtual]`

Set the object for the original filter line search.

Here, `orig_filter_ls_acceptor` must be the same strategy object to which the restoration phase object with this object is given. This method must be called to finish the definition of the algorithm, before `Initialize` is called.

Implements [Ipopt::RestoConvergenceCheck](#).

6.142.3.2 `virtual bool Ipopt::RestoFilterConvergenceCheck::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Reimplemented from [Ipopt::RestoConvergenceCheck](#).

6.142.3.3 `static void Ipopt::RestoFilterConvergenceCheck::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods used by `IpoptType`.

6.142.3.4 `void Ipopt::RestoFilterConvergenceCheck::operator= (const RestoFilterConvergenceCheck &) [private]`

Overloaded Equals Operator.

6.142.3.5 `virtual ConvergenceStatus Ipopt::RestoFilterConvergenceCheck::TestOrigProgress (Number orig_trial_barr, Number orig_trial_theta) [private],[virtual]`

Method for checking progress with original filter globalization mechanism.

Overloaded from [RestoConvergenceCheck](#).

Implements [Ipopt::RestoConvergenceCheck](#).

6.142.4 Member Data Documentation

6.142.4.1 `const FilterLSAcceptor* Ipopt::RestoFilterConvergenceCheck::orig_filter_ls_acceptor_ [private]`

Strategy object for the filter line search method for the original [NLP](#).

CAREFUL: We must not hold on to this object with a [SmartPtr](#), because have otherwise circular references that prevent the destructor of the line search object to be called!

Definition at line 77 of file `IpRestoFilterConvCheck.hpp`.

The documentation for this class was generated from the following file:

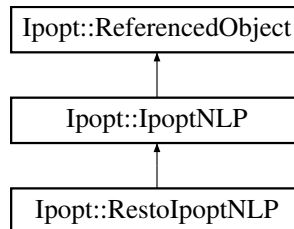
- [Algorithm/IpRestoFilterConvCheck.hpp](#)

6.143 Ipopt::RestoIpoptNLP Class Reference

This class maps the traditional [NLP](#) into something that is more useful by [Ipopt](#).

```
#include <IpRestoIpoptNLP.hpp>
```

Inheritance diagram for `Ipopt::RestoIpoptNLP`:



Public Member Functions

- virtual bool [Initialize](#) (const [Journalist](#) &jnlst, const [OptionsList](#) &options, const std::string &prefix)
Initialize - overloaded from [IpoptNLP](#).
- virtual bool [InitializeStructures](#) ([SmartPtr](#)< [Vector](#) > &x, bool init_x, [SmartPtr](#)< [Vector](#) > &y_c, bool init_y_c, [SmartPtr](#)< [Vector](#) > &y_d, bool init_y_d, [SmartPtr](#)< [Vector](#) > &z_L, bool init_z_L, [SmartPtr](#)< [Vector](#) > &z_U, bool init_z_U, [SmartPtr](#)< [Vector](#) > &v_L, [SmartPtr](#)< [Vector](#) > &v_U)
Initialize (create) structures for the iteration data.
- virtual bool [GetWarmStartIterate](#) ([IteratesVector](#) &warm_start_iterate)
Method accessing the [GetWarmStartIterate](#) of the [NLP](#).

- virtual void `GetSpaces` (`SmartPtr`< const `VectorSpace` > &x_space, `SmartPtr`< const `VectorSpace` > &c_space, `SmartPtr`< const `VectorSpace` > &d_space, `SmartPtr`< const `VectorSpace` > &x_l_space, `SmartPtr`< const `MatrixSpace` > &px_l_space, `SmartPtr`< const `VectorSpace` > &x_u_space, `SmartPtr`< const `MatrixSpace` > &px_u_space, `SmartPtr`< const `VectorSpace` > &d_l_space, `SmartPtr`< const `MatrixSpace` > &pd_l_space, `SmartPtr`< const `VectorSpace` > &d_u_space, `SmartPtr`< const `MatrixSpace` > &pd_u_space, `SmartPtr`< const `MatrixSpace` > &Jac_c_space, `SmartPtr`< const `MatrixSpace` > &Jac_d_space, `SmartPtr`< const `SymMatrixSpace` > &Hess_lagrangian_space)

Accessor method for vector/matrix spaces pointers.

- virtual void `AdjustVariableBounds` (const `Vector` &new_x_L, const `Vector` &new_x_U, const `Vector` &new_d_L, const `Vector` &new_d_U)

Method for adapting the variable bounds.

- bool `IntermediateCallBack` (`AlgorithmMode` mode, `Index` iter, `Number` obj_value, `Number` inf_pr, `Number` inf_du, `Number` mu, `Number` d_norm, `Number` regularization_size, `Number` alpha_du, `Number` alpha_pr, `Index` ls_trials, `SmartPtr`< const `IpoptData` > ip_data, `SmartPtr`< `IpoptCalculatedQuantities` > ip_cq)

User callback method.

- `Number` `Rho` () const

Accessor Method for obtaining the Rho penalization factor for the ell_1 norm.

- `Number` `Eta` (`Number` mu) const

Method to calculate eta, the factor for the regularization term.

- `SmartPtr`< const `Vector` > `DR_x` () const

Method returning the scaling factors for the 2-norm penalization term.

Constructors/Destructors

- `RestolpoptNLP` (`IpoptNLP` &orig_ip_nlp, `IpoptData` &orig_ip_data, `IpoptCalculatedQuantities` &orig_ip_cq)
- `~RestolpoptNLP` ()

Default destructor.

- void `FinalizeSolution` (`SolverReturn` status, const `Vector` &x, const `Vector` &z_L, const `Vector` &z_U, const `Vector` &c, const `Vector` &d, const `Vector` &y_c, const `Vector` &y_d, `Number` obj_value, const `IpoptData` *ip_data, `IpoptCalculatedQuantities` *ip_cq)

Solution Routines - overloaded from `IpoptNLP`.

- virtual bool `objective_depends_on_mu` () const

Accessor methods for model data.

- virtual `Number` `f` (const `Vector` &x)

Objective value (incorrect version for restoration phase)

- virtual `Number` `f` (const `Vector` &x, `Number` mu)

Objective value.

- virtual `SmartPtr`< const `Vector` > `grad_f` (const `Vector` &x)

Gradient of the objective (incorrect version for restoration phase)

- virtual `SmartPtr`< const `Vector` > `grad_f` (const `Vector` &x, `Number` mu)

Gradient of the objective.

- virtual `SmartPtr`< const `Vector` > `c` (const `Vector` &x)

Equality constraint residual.

- virtual `SmartPtr`< const `Matrix` > `jac_c` (const `Vector` &x)

Jacobian Matrix for equality constraints.

- virtual `SmartPtr`< const `Vector` > `d` (const `Vector` &x)

Inequality constraint residual (reformulated as equalities with slacks).

- virtual [SmartPtr](#)< const [Matrix](#) > [jac_d](#) (const [Vector](#) &x)
Jacobian [Matrix](#) for inequality constraints.
- virtual [SmartPtr](#)< const [SymMatrix](#) > [h](#) (const [Vector](#) &x, [Number](#) obj_factor, const [Vector](#) &yc, const [Vector](#) &yd)
Hessian of the Lagrangian (incorrect version for restoration phase)
- virtual [SmartPtr](#)< const [SymMatrix](#) > [h](#) (const [Vector](#) &x, [Number](#) obj_factor, const [Vector](#) &yc, const [Vector](#) &yd, [Number](#) mu)
Hessian of the Lagrangian.
- virtual [SmartPtr](#)< const [SymMatrix](#) > [uninitialized_h](#) ()
Provides a Hessian matrix from the correct matrix space with uninitialized values.
- virtual [SmartPtr](#)< const [Vector](#) > [x_L](#) () const
Lower bounds on x.
- virtual [SmartPtr](#)< const [Matrix](#) > [Px_L](#) () const
Permutation matrix ($x_L \rightarrow x$)
- virtual [SmartPtr](#)< const [Vector](#) > [x_U](#) () const
Upper bounds on x.
- virtual [SmartPtr](#)< const [Matrix](#) > [Px_U](#) () const
Permutation matrix ($x_U \rightarrow x$).
- virtual [SmartPtr](#)< const [Vector](#) > [d_L](#) () const
Lower bounds on d.
- virtual [SmartPtr](#)< const [Matrix](#) > [Pd_L](#) () const
Permutation matrix ($d_L \rightarrow d$)
- virtual [SmartPtr](#)< const [Vector](#) > [d_U](#) () const
Upper bounds on d.
- virtual [SmartPtr](#)< const [Matrix](#) > [Pd_U](#) () const
Permutation matrix ($d_U \rightarrow d$).
- virtual [SmartPtr](#)< const [SymMatrixSpace](#) > [HessianMatrixSpace](#) () const
Accessor method to obtain the [MatrixSpace](#) for the Hessian matrix (or it's approximation)

Accessor method for the information of the original NLP.

These methods are not overloaded from [IpoptNLP](#)

- [IpoptNLP](#) & [OrigIpNLP](#) () const
- [IpoptData](#) & [OrigIpData](#) () const
- [IpoptCalculatedQuantities](#) & [OrigIpCq](#) () const

Counters for the number of function evaluations.

- virtual [Index](#) [f_evals](#) () const
- virtual [Index](#) [grad_f_evals](#) () const
- virtual [Index](#) [c_evals](#) () const
- virtual [Index](#) [jac_c_evals](#) () const
- virtual [Index](#) [d_evals](#) () const
- virtual [Index](#) [jac_d_evals](#) () const
- virtual [Index](#) [h_evals](#) () const

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for [IpoptType](#).

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [RestolpoptNLP](#) ()
Default Constructor.
- [RestolpoptNLP](#) (const [RestolpoptNLP](#) &)
Copy Constructor.
- void [operator=](#) (const [RestolpoptNLP](#) &)
Overloaded Equals Operator.

Private Attributes

- bool [initialized_](#)
Flag indicating if initialization method has been called.

Pointers for the original NLP information.

- [SmartPtr](#)< [lpoptNLP](#) > [orig_ip_nlp_](#)
Pointer to the original [lpoptNLP](#).
- [SmartPtr](#)< [lpoptData](#) > [orig_ip_data_](#)
Pointer to the original [lpoptData](#).
- [SmartPtr](#)
< [lpoptCalculatedQuantities](#) > [orig_ip_cq_](#)
Pointer to the original [lpoptCalculatedQuantities](#).
- [SmartPtr](#)< [CompoundVectorSpace](#) > [x_space_](#)
Necessary Vector/Matrix spaces.
- [SmartPtr](#)< const [VectorSpace](#) > [c_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [d_space_](#)
- [SmartPtr](#)< [CompoundVectorSpace](#) > [x_l_space_](#)
- [SmartPtr](#)< [CompoundMatrixSpace](#) > [px_l_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [x_u_space_](#)
- [SmartPtr](#)< [CompoundMatrixSpace](#) > [px_u_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [d_l_space_](#)
- [SmartPtr](#)< const [MatrixSpace](#) > [pd_l_space_](#)
- [SmartPtr](#)< const [VectorSpace](#) > [d_u_space_](#)
- [SmartPtr](#)< const [MatrixSpace](#) > [pd_u_space_](#)
- [SmartPtr](#)< [CompoundMatrixSpace](#) > [jac_c_space_](#)
- [SmartPtr](#)< [CompoundMatrixSpace](#) > [jac_d_space_](#)
- [SmartPtr](#)< [CompoundSymMatrixSpace](#) > [h_space_](#)

Storage for Model Quantities

- [SmartPtr](#)< [CompoundVector](#) > [x_L_](#)
Lower bounds on x .
- [SmartPtr](#)< [CompoundMatrix](#) > [Px_L_](#)
Permutation matrix ($x_L \rightarrow x$)
- [SmartPtr](#)< const [Vector](#) > [x_U_](#)

- [SmartPtr< CompoundMatrix > Px_U_](#)
Upper bounds on x.
Permutation matrix ($x_{U_} \rightarrow x$)
- [SmartPtr< const Vector > d_L_](#)
Lower bounds on d.
- [SmartPtr< const Matrix > Pd_L_](#)
Permutation matrix ($d_{L_} \rightarrow d$)
- [SmartPtr< const Vector > d_U_](#)
Upper bounds on d.
- [SmartPtr< const Matrix > Pd_U_](#)
Permutation matrix ($d_{U_} \rightarrow d$)

Values particular to the restoration phase problem statement

- [Number rho_](#)
Penalty parameter for the $\| \cdot \|_1$ norm.
- [Number eta_factor_](#)
scaling factor for eta calculation
- [Number eta_mu_exponent_](#)
exponent for mu in eta calculation
- [SmartPtr< Vector > dr_x_](#)
Scaling factors for the $\$x\$$ part of the regularization term.
- [SmartPtr< DiagMatrix > DR_x_](#)
- [SmartPtr< Vector > x_ref_](#)
 $\$x\$$ part of the reference point in the regularization term

Algorithmic parameter

- [bool evaluate_orig_obj_at_resto_trial_](#)
Flag indicating if evaluation of the objective should be performed for every restoration phase objective function evaluation.
- [HessianApproximationType hessian_approximation_](#)
Flag indicating how hessian information is obtained.

Counters for the function evaluations

- [Index f_evals_](#)
- [Index grad_f_evals_](#)
- [Index c_evals_](#)
- [Index jac_c_evals_](#)
- [Index d_evals_](#)
- [Index jac_d_evals_](#)
- [Index h_evals_](#)

6.143.1 Detailed Description

This class maps the traditional [NLP](#) into something that is more useful by [Ipopt](#).

This class takes care of storing the calculated model results, handles cacheing, and (some day) takes care of addition of slacks.

Definition at line 32 of file IpRestIpoptNLP.hpp.

6.143.2 Constructor & Destructor Documentation

6.143.2.1 `Ipopt::RestIpoptNLP::RestIpoptNLP (IpoptNLP & orig_ip_nlp, IpoptData & orig_ip_data, IpoptCalculatedQuantities & orig_ip_cq)`

6.143.2.2 `Ipopt::RestIpoptNLP::~~RestIpoptNLP ()`

Default destructor.

6.143.2.3 `Ipopt::RestIpoptNLP::RestIpoptNLP () [private]`

Default Constructor.

6.143.2.4 `Ipopt::RestIpoptNLP::RestIpoptNLP (const RestIpoptNLP &) [private]`

Copy Constructor.

6.143.3 Member Function Documentation

6.143.3.1 `virtual bool Ipopt::RestIpoptNLP::Initialize (const Journalist & jnlst, const OptionsList & options, const std::string & prefix) [virtual]`

Initialize - overloaded from [IpoptNLP](#).

Reimplemented from [Ipopt::IpoptNLP](#).

6.143.3.2 `virtual bool Ipopt::RestIpoptNLP::InitializeStructures (SmartPtr< Vector > & x, bool init_x, SmartPtr< Vector > & y_c, bool init_y_c, SmartPtr< Vector > & y_d, bool init_y_d, SmartPtr< Vector > & z_L, bool init_z_L, SmartPtr< Vector > & z_U, bool init_z_U, SmartPtr< Vector > & v_L, SmartPtr< Vector > & v_U) [virtual]`

Initialize (create) structures for the iteration data.

Implements [Ipopt::IpoptNLP](#).

6.143.3.3 `virtual bool Ipopt::RestIpoptNLP::GetWarmStartIterate (IteratesVector & warm_start_iterate) [inline], [virtual]`

Method accessing the GetWarmStartIterate of the [NLP](#).

Implements [Ipopt::IpoptNLP](#).

Definition at line 67 of file `IpRestIpoptNLP.hpp`.

6.143.3.4 `void Ipopt::RestIpoptNLP::FinalizeSolution (SolverReturn status, const Vector & x, const Vector & z_L, const Vector & z_U, const Vector & c, const Vector & d, const Vector & y_c, const Vector & y_d, Number obj_value, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [inline], [virtual]`

Solution Routines - overloaded from [IpoptNLP](#).

Implements [Ipopt::IpoptNLP](#).

Definition at line 74 of file `IpRestIpoptNLP.hpp`.

6.143.3.5 `virtual bool Ipopt::RestIpoptNLP::objective_depends_on_mu () const [inline], [virtual]`

Accessor methods for model data.

Method for telling [IpoptCalculatedQuantities](#) that the restoration phase objective function depends on the barrier param-

eter

Reimplemented from [Ipopt::IpoptNLP](#).

Definition at line 89 of file IpRestIpoptNLP.hpp.

6.143.3.6 **virtual Number** Ipopt::RestIpoptNLP::f (const Vector & x) [virtual]

Objective value (incorrect version for restoration phase)

Implements [Ipopt::IpoptNLP](#).

6.143.3.7 **virtual Number** Ipopt::RestIpoptNLP::f (const Vector & x, Number mu) [virtual]

Objective value.

Implements [Ipopt::IpoptNLP](#).

6.143.3.8 **virtual SmartPtr<const Vector>** Ipopt::RestIpoptNLP::grad_f (const Vector & x) [virtual]

Gradient of the objective (incorrect version for restoration phase)

Implements [Ipopt::IpoptNLP](#).

6.143.3.9 **virtual SmartPtr<const Vector>** Ipopt::RestIpoptNLP::grad_f (const Vector & x, Number mu) [virtual]

Gradient of the objective.

Implements [Ipopt::IpoptNLP](#).

6.143.3.10 **virtual SmartPtr<const Vector>** Ipopt::RestIpoptNLP::c (const Vector & x) [virtual]

Equality constraint residual.

Implements [Ipopt::IpoptNLP](#).

6.143.3.11 **virtual SmartPtr<const Matrix>** Ipopt::RestIpoptNLP::jac_c (const Vector & x) [virtual]

Jacobian [Matrix](#) for equality constraints.

Implements [Ipopt::IpoptNLP](#).

6.143.3.12 **virtual SmartPtr<const Vector>** Ipopt::RestIpoptNLP::d (const Vector & x) [virtual]

Inequality constraint residual (reformulated as equalities with slacks).

Implements [Ipopt::IpoptNLP](#).

6.143.3.13 **virtual SmartPtr<const Matrix>** Ipopt::RestIpoptNLP::jac_d (const Vector & x) [virtual]

Jacobian [Matrix](#) for inequality constraints.

Implements [Ipopt::IpoptNLP](#).

6.143.3.14 **virtual SmartPtr<const SymMatrix>** Ipopt::RestIpoptNLP::h (const Vector & x, Number obj_factor, const Vector & yc, const Vector & yd) [virtual]

Hessian of the Lagrangian (incorrect version for restoration phase)

Implements [Ipopt::IpoptNLP](#).

6.143.3.15 `virtual SmartPtr<const SymMatrix> Ipopt::RestIpoptNLP::h (const Vector & x, Number obj_factor, const Vector & yc, const Vector & yd, Number mu) [virtual]`

Hessian of the Lagrangian.

Implements [Ipopt::IpoptNLP](#).

6.143.3.16 `virtual SmartPtr<const SymMatrix> Ipopt::RestIpoptNLP::uninitialized_h () [virtual]`

Provides a Hessian matrix from the correct matrix space with uninitialized values.

This can be used in LeastSquareMults to obtain a "zero Hessian".

Implements [Ipopt::IpoptNLP](#).

6.143.3.17 `virtual SmartPtr<const Vector> Ipopt::RestIpoptNLP::x_L () const [inline],[virtual]`

Lower bounds on x.

Implements [Ipopt::IpoptNLP](#).

Definition at line 140 of file IpRestIpoptNLP.hpp.

6.143.3.18 `virtual SmartPtr<const Matrix> Ipopt::RestIpoptNLP::Px_L () const [inline],[virtual]`

Permutation matrix (x_L_ -> x)

Implements [Ipopt::IpoptNLP](#).

Definition at line 146 of file IpRestIpoptNLP.hpp.

6.143.3.19 `virtual SmartPtr<const Vector> Ipopt::RestIpoptNLP::x_U () const [inline],[virtual]`

Upper bounds on x.

Implements [Ipopt::IpoptNLP](#).

Definition at line 152 of file IpRestIpoptNLP.hpp.

6.143.3.20 `virtual SmartPtr<const Matrix> Ipopt::RestIpoptNLP::Px_U () const [inline],[virtual]`

Permutation matrix (x_U_ -> x).

Implements [Ipopt::IpoptNLP](#).

Definition at line 158 of file IpRestIpoptNLP.hpp.

6.143.3.21 `virtual SmartPtr<const Vector> Ipopt::RestIpoptNLP::d_L () const [inline],[virtual]`

Lower bounds on d.

Implements [Ipopt::IpoptNLP](#).

Definition at line 164 of file IpRestIpoptNLP.hpp.

6.143.3.22 `virtual SmartPtr<const Matrix> Ipopt::RestIpoptNLP::Pd_L () const [inline],[virtual]`

Permutation matrix (d_L_ -> d)

Implements [Ipopt::IpoptNLP](#).

Definition at line 170 of file IpRestIpoptNLP.hpp.

6.143.3.23 `virtual SmartPtr<const Vector> Ipopt::RestIpoptNLP::d_U () const [inline],[virtual]`

Upper bounds on d.

Implements [Ipopt::IpoptNLP](#).

Definition at line 176 of file `IpRestIpoptNLP.hpp`.

6.143.3.24 `virtual SmartPtr<const Matrix> Ipopt::RestIpoptNLP::Pd_U () const [inline],[virtual]`

Permutation matrix (d_U_ -> d.

Implements [Ipopt::IpoptNLP](#).

Definition at line 182 of file `IpRestIpoptNLP.hpp`.

6.143.3.25 `virtual SmartPtr<const SymMatrixSpace> Ipopt::RestIpoptNLP::HessianMatrixSpace () const [inline],[virtual]`

Accessor method to obtain the [MatrixSpace](#) for the Hessian matrix (or it's approximation)

Implements [Ipopt::IpoptNLP](#).

Definition at line 187 of file `IpRestIpoptNLP.hpp`.

6.143.3.26 `virtual void Ipopt::RestIpoptNLP::GetSpaces (SmartPtr< const VectorSpace > & x_space, SmartPtr< const VectorSpace > & c_space, SmartPtr< const VectorSpace > & d_space, SmartPtr< const VectorSpace > & x_l_space, SmartPtr< const MatrixSpace > & px_l_space, SmartPtr< const VectorSpace > & x_u_space, SmartPtr< const MatrixSpace > & px_u_space, SmartPtr< const VectorSpace > & d_l_space, SmartPtr< const MatrixSpace > & pd_l_space, SmartPtr< const VectorSpace > & d_u_space, SmartPtr< const MatrixSpace > & pd_u_space, SmartPtr< const MatrixSpace > & Jac_c_space, SmartPtr< const MatrixSpace > & Jac_d_space, SmartPtr< const SymMatrixSpace > & Hess_lagrangian_space) [virtual]`

Accessor method for vector/matrix spaces pointers.

Implements [Ipopt::IpoptNLP](#).

6.143.3.27 `virtual void Ipopt::RestIpoptNLP::AdjustVariableBounds (const Vector & new_x_L, const Vector & new_x_U, const Vector & new_d_L, const Vector & new_d_U) [virtual]`

Method for adapting the variable bounds.

This is called if slacks are becoming too small

Implements [Ipopt::IpoptNLP](#).

6.143.3.28 `bool Ipopt::RestIpoptNLP::IntermediateCallBack (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, SmartPtr< const IpoptData > ip_data, SmartPtr< IpoptCalculatedQuantities > ip_cq) [virtual]`

User callback method.

Implements [Ipopt::IpoptNLP](#).

6.143.3.29 `IpoptNLP& Ipopt::RestIpoptNLP::OrigIpNLP () const [inline]`

Definition at line 229 of file `IpRestIpoptNLP.hpp`.

6.143.3.30 `IpoptData& Ipopt::RestolpoptNLP::OrigIpData () const [inline]`

Definition at line 233 of file IpRestolpoptNLP.hpp.

6.143.3.31 `IpoptCalculatedQuantities& Ipopt::RestolpoptNLP::OrigIpCq () const [inline]`

Definition at line 237 of file IpRestolpoptNLP.hpp.

6.143.3.32 `Number Ipopt::RestolpoptNLP::Rho () const [inline]`

Accessor Method for obtaining the Rho penalization factor for the ell_1 norm.

Definition at line 245 of file IpRestolpoptNLP.hpp.

6.143.3.33 `virtual Index Ipopt::RestolpoptNLP::f_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 252 of file IpRestolpoptNLP.hpp.

6.143.3.34 `virtual Index Ipopt::RestolpoptNLP::grad_f_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 256 of file IpRestolpoptNLP.hpp.

6.143.3.35 `virtual Index Ipopt::RestolpoptNLP::c_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 260 of file IpRestolpoptNLP.hpp.

6.143.3.36 `virtual Index Ipopt::RestolpoptNLP::jac_c_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 264 of file IpRestolpoptNLP.hpp.

6.143.3.37 `virtual Index Ipopt::RestolpoptNLP::d_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 268 of file IpRestolpoptNLP.hpp.

6.143.3.38 `virtual Index Ipopt::RestolpoptNLP::jac_d_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 272 of file IpRestolpoptNLP.hpp.

6.143.3.39 `virtual Index Ipopt::RestolpoptNLP::h_evals () const [inline],[virtual]`

Implements [Ipopt::IpoptNLP](#).

Definition at line 276 of file IpRestolpoptNLP.hpp.

6.143.3.40 `Number Ipopt::RestolpoptNLP::Eta (Number mu) const`

Method to calculate eta, the factor for the regularization term.

6.143.3.41 **SmartPtr<const Vector>** Ipopt::RestolpoptNLP::DR_x() const [inline]

Method returning the scaling factors for the 2-norm penalization term.

Definition at line 287 of file IpRestolpoptNLP.hpp.

6.143.3.42 **static void** Ipopt::RestolpoptNLP::RegisterOptions (**SmartPtr<RegisteredOptions>** *roptions*) [static]

Methods for IpoptType.

Called by IpoptType to register the options

6.143.3.43 **void** Ipopt::RestolpoptNLP::operator= (**const RestolpoptNLP &**) [private]

Overloaded Equals Operator.

6.143.4 Member Data Documentation

6.143.4.1 **SmartPtr<IpoptNLP>** Ipopt::RestolpoptNLP::orig_ip_nlp_ [private]

Pointer to the original [IpoptNLP](#).

Definition at line 302 of file IpRestolpoptNLP.hpp.

6.143.4.2 **SmartPtr<IpoptData>** Ipopt::RestolpoptNLP::orig_ip_data_ [private]

Pointer to the original [IpoptData](#).

Definition at line 305 of file IpRestolpoptNLP.hpp.

6.143.4.3 **SmartPtr<IpoptCalculatedQuantities>** Ipopt::RestolpoptNLP::orig_ip_cq_ [private]

Pointer to the original [IpoptCalculatedQuantities](#).

Definition at line 308 of file IpRestolpoptNLP.hpp.

6.143.4.4 **SmartPtr<CompoundVectorSpace>** Ipopt::RestolpoptNLP::x_space_ [private]

Necessary Vector/Matrix spaces.

Definition at line 313 of file IpRestolpoptNLP.hpp.

6.143.4.5 **SmartPtr<const VectorSpace>** Ipopt::RestolpoptNLP::c_space_ [private]

Definition at line 315 of file IpRestolpoptNLP.hpp.

6.143.4.6 **SmartPtr<const VectorSpace>** Ipopt::RestolpoptNLP::d_space_ [private]

Definition at line 317 of file IpRestolpoptNLP.hpp.

6.143.4.7 **SmartPtr<CompoundVectorSpace>** Ipopt::RestolpoptNLP::x_l_space_ [private]

Definition at line 319 of file IpRestolpoptNLP.hpp.

6.143.4.8 **SmartPtr<CompoundMatrixSpace>** Ipopt::RestolpoptNLP::px_l_space_ [private]

Definition at line 321 of file IpRestolpoptNLP.hpp.

6.143.4.9 **SmartPtr<const VectorSpace>** Ipopt::RestolpoptNLP::x_u_space_ [private]

Definition at line 323 of file IpRestolpoptNLP.hpp.

6.143.4.10 **SmartPtr<CompoundMatrixSpace>** Ipopt::RestolpoptNLP::px_u_space_ [private]

Definition at line 325 of file IpRestolpoptNLP.hpp.

6.143.4.11 **SmartPtr<const VectorSpace>** Ipopt::RestolpoptNLP::d_l_space_ [private]

Definition at line 327 of file IpRestolpoptNLP.hpp.

6.143.4.12 **SmartPtr<const MatrixSpace>** Ipopt::RestolpoptNLP::pd_l_space_ [private]

Definition at line 329 of file IpRestolpoptNLP.hpp.

6.143.4.13 **SmartPtr<const VectorSpace>** Ipopt::RestolpoptNLP::d_u_space_ [private]

Definition at line 331 of file IpRestolpoptNLP.hpp.

6.143.4.14 **SmartPtr<const MatrixSpace>** Ipopt::RestolpoptNLP::pd_u_space_ [private]

Definition at line 333 of file IpRestolpoptNLP.hpp.

6.143.4.15 **SmartPtr<CompoundMatrixSpace>** Ipopt::RestolpoptNLP::jac_c_space_ [private]

Definition at line 335 of file IpRestolpoptNLP.hpp.

6.143.4.16 **SmartPtr<CompoundMatrixSpace>** Ipopt::RestolpoptNLP::jac_d_space_ [private]

Definition at line 337 of file IpRestolpoptNLP.hpp.

6.143.4.17 **SmartPtr<CompoundSymMatrixSpace>** Ipopt::RestolpoptNLP::h_space_ [private]

Definition at line 339 of file IpRestolpoptNLP.hpp.

6.143.4.18 **SmartPtr<CompoundVector>** Ipopt::RestolpoptNLP::x_L_ [private]

Lower bounds on x.

Definition at line 345 of file IpRestolpoptNLP.hpp.

6.143.4.19 **SmartPtr<CompoundMatrix>** Ipopt::RestolpoptNLP::Px_L_ [private]

Permutation matrix (x_L_ -> x)

Definition at line 348 of file IpRestolpoptNLP.hpp.

6.143.4.20 **SmartPtr<const Vector>** Ipopt::RestolpoptNLP::x_U_ [private]

Upper bounds on x.

Definition at line 351 of file IpRestolpoptNLP.hpp.

6.143.4.21 **SmartPtr<CompoundMatrix>** Ipopt::RestolpoptNLP::Px_U_ [private]

Permutation matrix (x_U_ -> x)

Definition at line 354 of file IpRestolpoptNLP.hpp.

6.143.4.22 `SmartPtr<const Vector> Ipopt::RestolpoptNLP::d_L_ [private]`

Lower bounds on d.

Definition at line 357 of file IpRestolpoptNLP.hpp.

6.143.4.23 `SmartPtr<const Matrix> Ipopt::RestolpoptNLP::Pd_L_ [private]`

Permutation matrix (d_L_ -> d)

Definition at line 360 of file IpRestolpoptNLP.hpp.

6.143.4.24 `SmartPtr<const Vector> Ipopt::RestolpoptNLP::d_U_ [private]`

Upper bounds on d.

Definition at line 363 of file IpRestolpoptNLP.hpp.

6.143.4.25 `SmartPtr<const Matrix> Ipopt::RestolpoptNLP::Pd_U_ [private]`

Permutation matrix (d_U_ -> d).

Definition at line 366 of file IpRestolpoptNLP.hpp.

6.143.4.26 `Number Ipopt::RestolpoptNLP::rho_ [private]`

Penalty parameter for the $\| \cdot \|_1$ norm.

Definition at line 373 of file IpRestolpoptNLP.hpp.

6.143.4.27 `Number Ipopt::RestolpoptNLP::eta_factor_ [private]`

scaling factor for eta calculation

Definition at line 375 of file IpRestolpoptNLP.hpp.

6.143.4.28 `Number Ipopt::RestolpoptNLP::eta_mu_exponent_ [private]`

exponent for mu in eta calculation

Definition at line 377 of file IpRestolpoptNLP.hpp.

6.143.4.29 `SmartPtr<Vector> Ipopt::RestolpoptNLP::dr_x_ [private]`

Scaling factors for the λ part of the regularization term.

Definition at line 380 of file IpRestolpoptNLP.hpp.

6.143.4.30 `SmartPtr<DiagMatrix> Ipopt::RestolpoptNLP::DR_x_ [private]`

Definition at line 381 of file IpRestolpoptNLP.hpp.

6.143.4.31 `SmartPtr<Vector> Ipopt::RestolpoptNLP::x_ref_ [private]`

λ part of the reference point in the regularization term

Definition at line 383 of file IpRestolpoptNLP.hpp.

6.143.4.32 `bool Ipopt::RestolpoptNLP::evaluate_orig_obj_at_resto_trial_ [private]`

Flag indicating if evaluation of the objective should be performed for every restoration phase objective function evaluation.

Definition at line 409 of file IpRestolpoptNLP.hpp.

6.143.4.33 HessianApproximationType Ipopt::RestolpoptNLP::hessian_approximation_ [private]

Flag indicating how hessian information is obtained.

Definition at line 411 of file IpRestolpoptNLP.hpp.

6.143.4.34 bool Ipopt::RestolpoptNLP::initialized_ [private]

Flag indicating if initialization method has been called.

Definition at line 415 of file IpRestolpoptNLP.hpp.

6.143.4.35 Index Ipopt::RestolpoptNLP::f_evals_ [private]

Definition at line 419 of file IpRestolpoptNLP.hpp.

6.143.4.36 Index Ipopt::RestolpoptNLP::grad_f_evals_ [private]

Definition at line 420 of file IpRestolpoptNLP.hpp.

6.143.4.37 Index Ipopt::RestolpoptNLP::c_evals_ [private]

Definition at line 421 of file IpRestolpoptNLP.hpp.

6.143.4.38 Index Ipopt::RestolpoptNLP::jac_c_evals_ [private]

Definition at line 422 of file IpRestolpoptNLP.hpp.

6.143.4.39 Index Ipopt::RestolpoptNLP::d_evals_ [private]

Definition at line 423 of file IpRestolpoptNLP.hpp.

6.143.4.40 Index Ipopt::RestolpoptNLP::jac_d_evals_ [private]

Definition at line 424 of file IpRestolpoptNLP.hpp.

6.143.4.41 Index Ipopt::RestolpoptNLP::h_evals_ [private]

Definition at line 425 of file IpRestolpoptNLP.hpp.

The documentation for this class was generated from the following file:

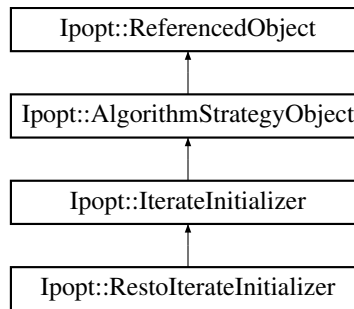
- [Algorithm/IpRestolpoptNLP.hpp](#)

6.144 Ipopt::RestolterateInitializer Class Reference

Class implementing the default initialization procedure (based on user options) for the iterates.

```
#include <IpRestoIterateInitializer.hpp>
```

Inheritance diagram for Ipopt::RestolterateInitializer:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [SetInitialIterates](#) ()
Compute the initial iterates and set the into the curr field of the ip_data object.

Constructors/Destructors

- [RestolterateInitializer](#) (const [SmartPtr](#)< [EqMultiplierCalculator](#) > &eq_mult_calculator)
Constructor.
- virtual [~RestolterateInitializer](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [RestolterateInitializer](#) ()
Default Constructor.
- [RestolterateInitializer](#) (const [RestolterateInitializer](#) &)
Copy Constructor.
- void [operator=](#) (const [RestolterateInitializer](#) &)
Overloaded Equals Operator.

Auxilliary functions

- void [solve_quadratic](#) (const [Vector](#) &a, const [Vector](#) &b, [Vector](#) &v)
*Method for solving the quadratic vector equation $v^2 + 2a*v - b = 0$*

Private Attributes

- [SmartPtr< EqMultiplierCalculator > resto_eq_mult_calculator_](#)
object to be used for the initialization of the equality constraint multipliers.

Parameters for bumping x0

- [Number constr_mult_init_max_](#)
If max-norm of the initial equality constraint multiplier estimate is larger than this, the initial y_ variables are set to zero.*

Additional Inherited Members

6.144.1 Detailed Description

Class implementing the default initialization procedure (based on user options) for the iterates.

It is used at the very beginning of the optimization for determine the starting point for all variables.

Definition at line 22 of file IpRestolterateInitializer.hpp.

6.144.2 Constructor & Destructor Documentation

6.144.2.1 `Ipopt::RestolterateInitializer::RestolterateInitializer (const SmartPtr< EqMultiplierCalculator > & eq_mult_calculator)`

Constructor.

If eq_mult_calculator is not NULL, it will be used to compute the initial values for equality constraint multipliers.

6.144.2.2 `virtual Ipopt::RestolterateInitializer::~RestolterateInitializer () [inline], [virtual]`

Default destructor.

Definition at line 34 of file IpRestolterateInitializer.hpp.

6.144.2.3 `Ipopt::RestolterateInitializer::RestolterateInitializer () [private]`

Default Constructor.

6.144.2.4 `Ipopt::RestolterateInitializer::RestolterateInitializer (const RestolterateInitializer &) [private]`

Copy Constructor.

6.144.3 Member Function Documentation

6.144.3.1 `virtual bool Ipopt::RestolterateInitializer::InitializImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::IterateInitializer](#).

6.144.3.2 `virtual bool Ipopt::RestolterateInitializer::SetInitialIterates () [virtual]`

Compute the initial iterates and set the into the curr field of the ip_data object.

Implements [Ipopt::IterateInitializer](#).

6.144.3.3 `static void Ipopt::RestolteratnInitializer::RegisterOptions (SmartPtr< RegisteredOptions > options)`
`[static]`

Methods for IpoptType.

6.144.3.4 `void Ipopt::RestolteratnInitializer::operator= (const RestolteratnInitializer &)` `[private]`

Overloaded Equals Operator.

6.144.3.5 `void Ipopt::RestolteratnInitializer::solve_quadratic (const Vector & a, const Vector & b, Vector & v)` `[private]`

Method for solving the quadratic vector equation $v^2 + 2a*v -$

$b = 0$

6.144.4 Member Data Documentation

6.144.4.1 `Number Ipopt::RestolteratnInitializer::constr_mult_init_max_` `[private]`

If max-norm of the initial equality constraint multiplier estimate is larger than this, the initial y_* variables are set to zero.

Definition at line 75 of file `IpRestolteratnInitializer.hpp`.

6.144.4.2 `SmartPtr<EqMultiplierCalculator> Ipopt::RestolteratnInitializer::resto_eq_mult_calculator_` `[private]`

object to be used for the initialization of the equality constraint multipliers.

Definition at line 80 of file `IpRestolteratnInitializer.hpp`.

The documentation for this class was generated from the following file:

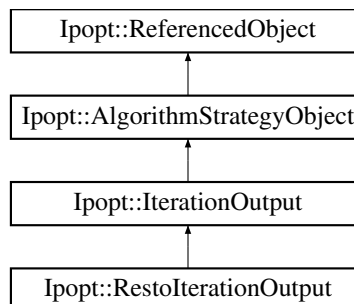
- [Algorithm/IpRestolteratnInitializer.hpp](#)

6.145 Ipopt::RestolterationOutput Class Reference

Class for the iteration summary output for the restoration phase.

`#include <IpRestoIterationOutput.hpp>`

Inheritance diagram for `Ipopt::RestolterationOutput`:



Public Member Functions

- virtual bool `InitializeImpl` (const [OptionsList](#) &options, const std::string &prefix)

overloaded from [AlgorithmStrategyObject](#)

- virtual void [WriteOutput](#) ()

Method to do all the summary output per iteration.

Constructors/Destructors

- [RestolterationOutput](#) (const [SmartPtr](#)< [OrigIterationOutput](#) > &resto_orig_iteration_output)

Constructor.

- virtual [~RestolterationOutput](#) ()

Default destructor.

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [RestolterationOutput](#) ()
Default Constructor.
- [RestolterationOutput](#) (const [RestolterationOutput](#) &)
Copy Constructor.
- void [operator=](#) (const [RestolterationOutput](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [OrigIterationOutput](#) > [resto_orig_iteration_output_](#)
Pointer to output strategy object during regular iterations.
- bool [print_info_string_](#)
Flag indicating weather info string should be printed at end of iteration summary line.
- [InfPrOutput](#) [inf_pr_output_](#)
Option indication what should be printed in inf_pr column.
- int [print_frequency_iter_](#)
Option indicating at which iteration frequency the summary line should be printed.
- Number [print_frequency_time_](#)
Option indicating at which time frequency the summary line should be printed.

Additional Inherited Members

6.145.1 Detailed Description

Class for the iteration summary output for the restoration phase.

This prints information for the ORIGINAL [NLP](#) (and possibly for the restoration phase [NLP](#)).

Definition at line 22 of file [IpRestolterationOutput.hpp](#).

6.145.2 Constructor & Destructor Documentation

6.145.2.1 `Ipopt::RestolterationOutput::RestolterationOutput (const SmartPtr< OrigIterationOutput > & resto_orig_iteration_output)`

Constructor.

If `resto_orig_iteration_output` is not NULL, the output will be done twice per iteration, first for the restoration phase problem, and secondly using the functions for the original [NLP](#).

6.145.2.2 `virtual Ipopt::RestolterationOutput::~~RestolterationOutput () [virtual]`

Default destructor.

6.145.2.3 `Ipopt::RestolterationOutput::RestolterationOutput () [private]`

Default Constructor.

6.145.2.4 `Ipopt::RestolterationOutput::RestolterationOutput (const RestolterationOutput &) [private]`

Copy Constructor.

6.145.3 Member Function Documentation

6.145.3.1 `virtual bool Ipopt::RestolterationOutput::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::IterationOutput](#).

6.145.3.2 `virtual void Ipopt::RestolterationOutput::WriteOutput () [virtual]`

Method to do all the summary output per iteration.

This include the one-line summary output as well as writing the details about the iterates if desired

Implements [Ipopt::IterationOutput](#).

6.145.3.3 `void Ipopt::RestolterationOutput::operator= (const RestolterationOutput &) [private]`

Overloaded Equals Operator.

6.145.4 Member Data Documentation

6.145.4.1 `SmartPtr<OrigIterationOutput> Ipopt::RestolterationOutput::resto_orig_iteration_output_ [private]`

Pointer to output strategy object during regular iterations.

Definition at line 64 of file `IpRestolterationOutput.hpp`.

6.145.4.2 `bool Ipopt::RestolterationOutput::print_info_string_ [private]`

Flag indicating weather info string should be printed at end of iteration summary line.

Definition at line 68 of file `IpRestolterationOutput.hpp`.

6.145.4.3 InfPrOutput Ipopt::RestolterationOutput::inf_pr_output_ [private]

Option indication what should be printed in inf_pr column.

Definition at line 71 of file IpRestolterationOutput.hpp.

6.145.4.4 int Ipopt::RestolterationOutput::print_frequency_iter_ [private]

Option indicating at which iteration frequency the summary line should be printed.

Definition at line 74 of file IpRestolterationOutput.hpp.

6.145.4.5 Number Ipopt::RestolterationOutput::print_frequency_time_ [private]

Option indicating at which time frequency the summary line should be printed.

Definition at line 76 of file IpRestolterationOutput.hpp.

The documentation for this class was generated from the following file:

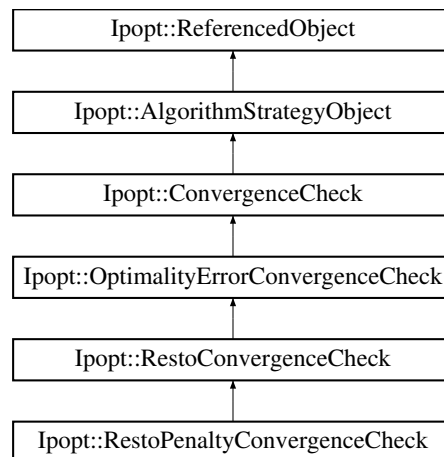
- [Algorithm/IpRestolterationOutput.hpp](#)

6.146 Ipopt::RestoPenaltyConvergenceCheck Class Reference

This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism.

```
#include <IpRestoPenaltyConvCheck.hpp>
```

Inheritance diagram for Ipopt::RestoPenaltyConvergenceCheck:



Public Member Functions

- void [SetOrigLSAcceptor](#) (const [BacktrackingLSAcceptor](#) &orig_ls_acceptor)
Set the object for the original penalty line search.
- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)

Constructors/Destructors

- [RestoPenaltyConvergenceCheck](#) ()
Default Constructor.
- virtual [~RestoPenaltyConvergenceCheck](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) (SmartPtr< [RegisteredOptions](#) > roptions)
Methods used by IpoptType.

Private Member Functions

- virtual [ConvergenceStatus TestOrigProgress](#) (Number orig_trial_barr, Number orig_trial_theta)
Method for checking progress with original filter globalization mechanism.

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [RestoPenaltyConvergenceCheck](#) (const [RestoPenaltyConvergenceCheck](#) &)
Copy Constructor.
- void [operator=](#) (const [RestoPenaltyConvergenceCheck](#) &)
Overloaded Equals Operator.

Private Attributes

- const [PenaltyLSAcceptor](#) * [orig_penalty_ls_acceptor_](#)
Strategy object for the filter line search method for the original NLP.

Additional Inherited Members

6.146.1 Detailed Description

This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism.

Definition at line 23 of file IpRestoPenaltyConvCheck.hpp.

6.146.2 Constructor & Destructor Documentation

6.146.2.1 Ipopt::RestoPenaltyConvergenceCheck::RestoPenaltyConvergenceCheck ()

Default Constructor.

6.146.2.2 virtual Ipopt::RestoPenaltyConvergenceCheck::~~RestoPenaltyConvergenceCheck () [virtual]

Default destructor.

6.146.2.3 `Ipopt::RestoPenaltyConvergenceCheck::RestoPenaltyConvergenceCheck (const RestoPenaltyConvergenceCheck &) [private]`

Copy Constructor.

6.146.3 Member Function Documentation

6.146.3.1 `void Ipopt::RestoPenaltyConvergenceCheck::SetOrigLSAcceptor (const BacktrackingLSAcceptor & orig_ls_acceptor) [virtual]`

Set the object for the original penalty line search.

Here, `orig_penalty_ls_acceptor` must be the same strategy object to which the restoration phase object with this object is given. This method must be called to finish the definition of the algorithm, before `Initialize` is called.

Implements [Ipopt::RestoConvergenceCheck](#).

6.146.3.2 `virtual bool Ipopt::RestoPenaltyConvergenceCheck::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Reimplemented from [Ipopt::RestoConvergenceCheck](#).

6.146.3.3 `static void Ipopt::RestoPenaltyConvergenceCheck::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods used by `IpoptType`.

6.146.3.4 `void Ipopt::RestoPenaltyConvergenceCheck::operator= (const RestoPenaltyConvergenceCheck &) [private]`

Overloaded Equals Operator.

6.146.3.5 `virtual ConvergenceStatus Ipopt::RestoPenaltyConvergenceCheck::TestOrigProgress (Number orig_trial_barr, Number orig_trial_theta) [private],[virtual]`

Method for checking progress with original filter globalization mechanism.

Overloaded from [RestoConvergenceCheck](#).

Implements [Ipopt::RestoConvergenceCheck](#).

6.146.4 Member Data Documentation

6.146.4.1 `const PenaltyLSAcceptor* Ipopt::RestoPenaltyConvergenceCheck::orig_penalty_ls_acceptor_ [private]`

Strategy object for the filter line search method for the original [NLP](#).

CAREFUL: We must not hold on to this object with a [SmartPtr](#), because have otherwise circular references that prevent the destructor of the line search object to be called!

Definition at line 76 of file `IpRestoPenaltyConvCheck.hpp`.

The documentation for this class was generated from the following file:

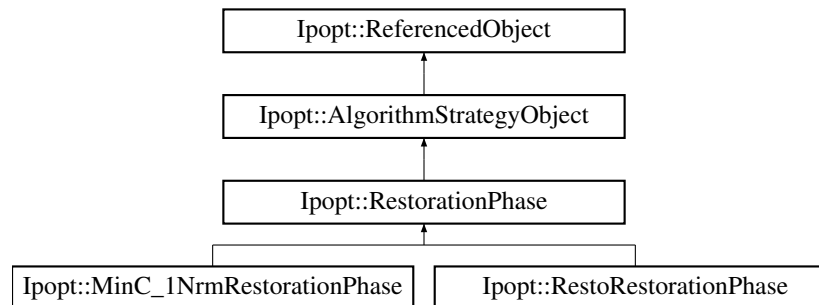
- [Algorithm/IpRestoPenaltyConvCheck.hpp](#)

6.147 Ipopt::RestorationPhase Class Reference

Base class for different restoration phases.

```
#include <IpRestoPhase.hpp>
```

Inheritance diagram for Ipopt::RestorationPhase:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [PerformRestoration](#) ()=0
Method called to perform restoration for the filter line search method.

Constructors/Destructors

- [RestorationPhase](#) ()
Default Constructor.
- virtual [~RestorationPhase](#) ()
Default Destructor.

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [RestorationPhase](#) (const [RestorationPhase](#) &)
Copy Constructor.
- void [operator=](#) (const [RestorationPhase](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.147.1 Detailed Description

Base class for different restoration phases.

The restoration phase is part of the FilterLineSearch.

Definition at line 34 of file IpRestoPhase.hpp.

6.147.2 Constructor & Destructor Documentation

6.147.2.1 `Ipopt::RestorationPhase::RestorationPhase ()` `[inline]`

Default Constructor.

Definition at line 40 of file `IpRestoPhase.hpp`.

6.147.2.2 `virtual Ipopt::RestorationPhase::~~RestorationPhase ()` `[inline]`, `[virtual]`

Default Destructor.

Definition at line 43 of file `IpRestoPhase.hpp`.

6.147.2.3 `Ipopt::RestorationPhase::RestorationPhase (const RestorationPhase &)` `[private]`

Copy Constructor.

6.147.3 Member Function Documentation

6.147.3.1 `virtual bool Ipopt::RestorationPhase::InitializeImpl (const OptionsList & options, const std::string & prefix)` `[pure virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::MinC_1NrmRestorationPhase](#), and [Ipopt::RestoRestorationPhase](#).

6.147.3.2 `virtual bool Ipopt::RestorationPhase::PerformRestoration ()` `[pure virtual]`

Method called to perform restoration for the filter line search method.

Implemented in [Ipopt::MinC_1NrmRestorationPhase](#), and [Ipopt::RestoRestorationPhase](#).

6.147.3.3 `void Ipopt::RestorationPhase::operator= (const RestorationPhase &)` `[private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

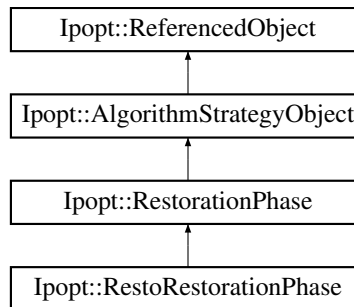
- [Algorithm/IpRestoPhase.hpp](#)

6.148 `Ipopt::RestoRestorationPhase` Class Reference

Recursive Restoration Phase for the `MinC_1NrmRestorationPhase`.

```
#include <IpRestoRestoPhase.hpp>
```

Inheritance diagram for `Ipopt::RestoRestorationPhase`:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
Overloaded from [AlgorithmStrategy](#) case class.

Constructors/Destructors

- [RestoRestorationPhase](#) ()
Default Constructor.
- virtual [~RestoRestorationPhase](#) ()
Default destructor.

Protected Member Functions

- virtual bool [PerformRestoration](#) ()
Overloaded method from [RestorationPhase](#).

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [RestoRestorationPhase](#) (const [RestoRestorationPhase](#) &)
Copy Constructor.
- void [operator=](#) (const [RestoRestorationPhase](#) &)
Overloaded Equals Operator.

Auxilliary methods

- void [solve_quadratic](#) (const [Vector](#) &a, const [Vector](#) &b, [Vector](#) &v)
*Method for solving the quadratic vector equation $v^2 + 2a*v - b = 0$*

6.148.1 Detailed Description

Recursive Restoration Phase for the [MinC_1NrmRestorationPhase](#).

This procedure chooses the n and p variables in the [MinC_1NrmRestorationPhase](#) problem formulation by treating the problem as separable (assuming that the x and s variables are fixed).

Definition at line 25 of file [IpRestoRestoPhase.hpp](#).

6.148.2 Constructor & Destructor Documentation

6.148.2.1 `Ipopt::RestoRestorationPhase::RestoRestorationPhase ()`

Default Constructor.

6.148.2.2 `virtual Ipopt::RestoRestorationPhase::~~RestoRestorationPhase () [virtual]`

Default destructor.

6.148.2.3 `Ipopt::RestoRestorationPhase::RestoRestorationPhase (const RestoRestorationPhase &) [private]`

Copy Constructor.

6.148.3 Member Function Documentation

6.148.3.1 `virtual bool Ipopt::RestoRestorationPhase::InitializImpl (const OptionsList & options, const std::string & prefix) [virtual]`

Overloaded from AlgorithmStrategy case class.

Implements [Ipopt::RestorationPhase](#).

6.148.3.2 `virtual bool Ipopt::RestoRestorationPhase::PerformRestoration () [protected],[virtual]`

Overloaded method from [RestorationPhase](#).

Implements [Ipopt::RestorationPhase](#).

6.148.3.3 `void Ipopt::RestoRestorationPhase::operator= (const RestoRestorationPhase &) [private]`

Overloaded Equals Operator.

6.148.3.4 `void Ipopt::RestoRestorationPhase::solve_quadratic (const Vector & a, const Vector & b, Vector & v) [private]`

Method for solving the quadratic vector equation $v^2 + 2a*v -$

$b = 0$

The documentation for this class was generated from the following file:

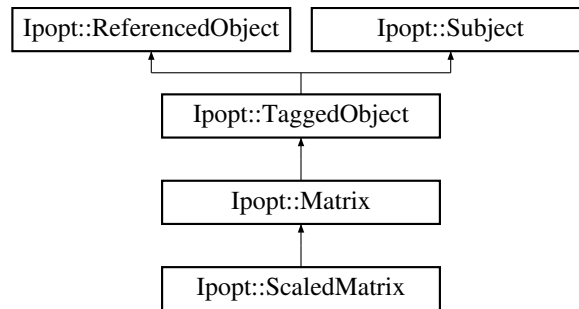
- [Algorithm/IpRestoRestoPhase.hpp](#)

6.149 Ipopt::ScaledMatrix Class Reference

Class for a [Matrix](#) in conjunction with its scaling factors for row and column scaling.

```
#include <IpScaledMatrix.hpp>
```

Inheritance diagram for Ipopt::ScaledMatrix:



Public Member Functions

- void [SetUnscaledMatrix](#) (const [SmartPtr](#)< const [Matrix](#) > unscaled_matrix)
Set the unscaled matrix.
- void [SetUnscaledMatrixNonConst](#) (const [SmartPtr](#)< [Matrix](#) > &unscaled_matrix)
Set the unscaled matrix in a non-const version.
- [SmartPtr](#)< const [Matrix](#) > [GetUnscaledMatrix](#) () const
Return the unscaled matrix in const form.
- [SmartPtr](#)< [Matrix](#) > [GetUnscaledMatrixNonConst](#) ()
Return the unscaled matrix in non-const form.
- [SmartPtr](#)< const [Vector](#) > [RowScaling](#) () const
return the vector for the row scaling
- [SmartPtr](#)< const [Vector](#) > [ColumnScaling](#) () const
return the vector for the column scaling

Constructors / Destructors

- [ScaledMatrix](#) (const [ScaledMatrixSpace](#) *owner_space)
Constructor, taking the owner_space.
- [~ScaledMatrix](#) ()
Destructor.

Protected Member Functions

Methods overloaded from Matrix

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual void [TransMultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix(transpose) vector multiply.
- virtual bool [IsValidNumbersImpl](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &cols_norms, bool init) const
Compute the max-norm of the columns in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.
- virtual void [AddMSinvZImpl](#) ([Number](#) alpha, const [Vector](#) &S, const [Vector](#) &Z, [Vector](#) &X) const

- $$X = \text{beta} * X + \text{alpha} * (\text{Matrix } S^{-1} \{ -1 \} Z).$$
- virtual void `SinvBlrmZMTdBrlImpl` (Number alpha, const `Vector` &S, const `Vector` &R, const `Vector` &Z, const `Vector` &D, `Vector` &X) const
- $$X = S^{-1} \{ -1 \} (r + \text{alpha} * Z * M^{\wedge} Td).$$

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `ScaledMatrix` ()
Default Constructor.
- `ScaledMatrix` (const `ScaledMatrix` &)
Copy Constructor.
- void `operator=` (const `ScaledMatrix` &)
Overloaded Equals Operator.

Private Attributes

- `SmartPtr`< const `Matrix` > `matrix_`
const version of the unscaled matrix
- `SmartPtr`< `Matrix` > `nonconst_matrix_`
non-const version of the unscaled matrix
- `SmartPtr`< const `ScaledMatrixSpace` > `owner_space_`
`Matrix` space stored as a `ScaledMatrixSpace`.

Additional Inherited Members

6.149.1 Detailed Description

Class for a `Matrix` in conjunction with its scaling factors for row and column scaling.

Operations on the matrix are performed using the scaled matrix. You can pull out the pointer to the unscaled matrix for unscaled calculations.

Definition at line 26 of file `IpScaledMatrix.hpp`.

6.149.2 Constructor & Destructor Documentation

6.149.2.1 `Ipopt::ScaledMatrix::ScaledMatrix (const ScaledMatrixSpace * owner_space)`

Constructor, taking the `owner_space`.

6.149.2.2 `Ipopt::ScaledMatrix::~~ScaledMatrix ()`

Destructor.

6.149.2.3 `Ipopt::ScaledMatrix::ScaledMatrix () [private]`

Default Constructor.

6.149.2.4 `Ipopt::ScaledMatrix::ScaledMatrix (const ScaledMatrix &) [private]`

Copy Constructor.

6.149.3 Member Function Documentation

6.149.3.1 `void Ipopt::ScaledMatrix::SetUnscaledMatrix (const SmartPtr< const Matrix > unscaled_matrix) [inline]`

Set the unscaled matrix.

Definition at line 211 of file IpScaledMatrix.hpp.

6.149.3.2 `void Ipopt::ScaledMatrix::SetUnscaledMatrixNonConst (const SmartPtr< Matrix > & unscaled_matrix) [inline]`

Set the unscaled matrix in a non-const version.

Definition at line 219 of file IpScaledMatrix.hpp.

6.149.3.3 `SmartPtr< const Matrix > Ipopt::ScaledMatrix::GetUnscaledMatrix () const [inline]`

Return the unscaled matrix in const form.

Definition at line 227 of file IpScaledMatrix.hpp.

6.149.3.4 `SmartPtr< Matrix > Ipopt::ScaledMatrix::GetUnscaledMatrixNonConst () [inline]`

Return the unscaled matrix in non-const form.

Definition at line 233 of file IpScaledMatrix.hpp.

6.149.3.5 `SmartPtr< const Vector > Ipopt::ScaledMatrix::RowScaling () const [inline]`

return the vector for the row scaling

Definition at line 241 of file IpScaledMatrix.hpp.

6.149.3.6 `SmartPtr< const Vector > Ipopt::ScaledMatrix::ColumnScaling () const [inline]`

return the vector for the column scaling

Definition at line 247 of file IpScaledMatrix.hpp.

6.149.3.7 `virtual void Ipopt::ScaledMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.149.3.8 `virtual void Ipopt::ScaledMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.149.3.9 `virtual bool Ipopt::ScaledMatrix::IsValidNumbersImpl () const` [protected], [virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

It is assumed that the scaling factors are valid.

Reimplemented from [Ipopt::Matrix](#).

6.149.3.10 `virtual void Ipopt::ScaledMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` [protected], [virtual]

Compute the max-norm of the rows in the matrix.

The result is stored in `rows_norms`. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.149.3.11 `virtual void Ipopt::ScaledMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const` [protected], [virtual]

Compute the max-norm of the columns in the matrix.

The result is stored in `cols_norms`. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.149.3.12 `virtual void Ipopt::ScaledMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected], [virtual]

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.149.3.13 `virtual void Ipopt::ScaledMatrix::AddMSinvZImpl (Number alpha, const Vector & S, const Vector & Z, Vector & X) const` [protected], [virtual]

$X = \beta X + \alpha (S^{-1} Z)$.

Specialized implementation missing so far!

Reimplemented from [Ipopt::Matrix](#).

6.149.3.14 `virtual void Ipopt::ScaledMatrix::SinvBlrmZMTdBrlmImpl (Number alpha, const Vector & S, const Vector & R, const Vector & Z, const Vector & D, Vector & X) const` [protected], [virtual]

$X = S^{-1} (r + \alpha Z M^T D)$.

Specialized implementation missing so far!

Reimplemented from [Ipopt::Matrix](#).

6.149.3.15 `void Ipopt::ScaledMatrix::operator= (const ScaledMatrix &)` [private]

Overloaded Equals Operator.

6.149.4 Member Data Documentation

6.149.4.1 `SmartPtr<const Matrix> Ipopt::ScaledMatrix::matrix_` [private]

const version of the unscaled matrix

Definition at line 118 of file IpScaledMatrix.hpp.

6.149.4.2 **SmartPointer<Matrix>** Ipopt::ScaledMatrix::nonconst_matrix_ [private]

non-const version of the unscaled matrix

Definition at line 120 of file IpScaledMatrix.hpp.

6.149.4.3 **SmartPointer<const ScaledMatrixSpace>** Ipopt::ScaledMatrix::owner_space_ [private]

Matrix space stored as a ScaledMatrixSpace.

Definition at line 123 of file IpScaledMatrix.hpp.

The documentation for this class was generated from the following file:

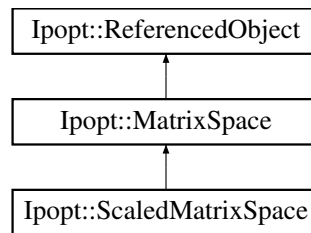
- LinAlg/IpScaledMatrix.hpp

6.150 Ipopt::ScaledMatrixSpace Class Reference

This is the matrix space for ScaledMatrix.

```
#include <IpScaledMatrix.hpp>
```

Inheritance diagram for Ipopt::ScaledMatrixSpace:



Public Member Functions

- **ScaledMatrix * MakeNewScaledMatrix** (bool allocate_unscaled_matrix=false) const
Method for creating a new matrix of this specific type.
- virtual **Matrix * MakeNew** () const
Overloaded MakeNew method for the MatrixSpace base class.
- **SmartPointer< const Vector > RowScaling** () const
return the vector for the row scaling
- **SmartPointer< const MatrixSpace > UnscaledMatrixSpace** () const
return the matrix space for the unscaled matrix
- **SmartPointer< const Vector > ColumnScaling** () const
return the vector for the column scaling

Constructors / Destructors

- **ScaledMatrixSpace** (const SmartPtr< const Vector > &row_scaling, bool row_scaling_reciprocal, const SmartPtr< const MatrixSpace > &unscaled_matrix_space, const SmartPtr< const Vector > &column_scaling, bool column_scaling_reciprocal)
Constructor, given the number of row and columns blocks, as well as the total number of rows and columns.
- **~ScaledMatrixSpace** ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ScaledMatrixSpace](#) ()
Default constructor.
- [ScaledMatrixSpace](#) (const [ScaledMatrixSpace](#) &)
Copy Constructor.
- [ScaledMatrixSpace](#) & [operator=](#) (const [ScaledMatrixSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [Vector](#) > [row_scaling_](#)
Row scaling vector.
- [SmartPtr](#)< const [MatrixSpace](#) > [unscaled_matrix_space_](#)
unscaled matrix space
- [SmartPtr](#)< [Vector](#) > [column_scaling_](#)
column scaling vector

6.150.1 Detailed Description

This is the matrix space for [ScaledMatrix](#).

Definition at line 128 of file [IpScaledMatrix.hpp](#).

6.150.2 Constructor & Destructor Documentation

6.150.2.1 [Ipopt::ScaledMatrixSpace::ScaledMatrixSpace](#) (const [SmartPtr](#)< const [Vector](#) > & [row_scaling](#), bool [row_scaling_reciprocal](#), const [SmartPtr](#)< const [MatrixSpace](#) > & [unscaled_matrix_space](#), const [SmartPtr](#)< const [Vector](#) > & [column_scaling](#), bool [column_scaling_reciprocal](#))

Constructor, given the number of row and columns blocks, as well as the total number of rows and columns.

6.150.2.2 [Ipopt::ScaledMatrixSpace::~~ScaledMatrixSpace](#) () [inline]

Destructor.

Definition at line 143 of file [IpScaledMatrix.hpp](#).

6.150.2.3 [Ipopt::ScaledMatrixSpace::ScaledMatrixSpace](#) () [private]

Default constructor.

6.150.2.4 [Ipopt::ScaledMatrixSpace::ScaledMatrixSpace](#) (const [ScaledMatrixSpace](#) &) [private]

Copy Constructor.

6.150.3 Member Function Documentation

6.150.3.1 `ScaledMatrix* Ipopt::ScaledMatrixSpace::MakeNewScaledMatrix (bool allocate_unscaled_matrix = false) const` `[inline]`

Method for creating a new matrix of this specific type.

Definition at line 148 of file IpScaledMatrix.hpp.

6.150.3.2 `virtual Matrix* Ipopt::ScaledMatrixSpace::MakeNew () const` `[inline], [virtual]`

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 160 of file IpScaledMatrix.hpp.

6.150.3.3 `SmartPtr<const Vector> Ipopt::ScaledMatrixSpace::RowScaling () const` `[inline]`

return the vector for the row scaling

Definition at line 166 of file IpScaledMatrix.hpp.

6.150.3.4 `SmartPtr<const MatrixSpace> Ipopt::ScaledMatrixSpace::UnscaledMatrixSpace () const` `[inline]`

return the matrix space for the unscaled matrix

Definition at line 172 of file IpScaledMatrix.hpp.

6.150.3.5 `SmartPtr<const Vector> Ipopt::ScaledMatrixSpace::ColumnScaling () const` `[inline]`

return the vector for the column scaling

Definition at line 178 of file IpScaledMatrix.hpp.

6.150.3.6 `ScaledMatrixSpace& Ipopt::ScaledMatrixSpace::operator= (const ScaledMatrixSpace &)` `[private]`

Overloaded Equals Operator.

6.150.4 Member Data Documentation

6.150.4.1 `SmartPtr<Vector> Ipopt::ScaledMatrixSpace::row_scaling_` `[private]`

Row scaling vector.

Definition at line 203 of file IpScaledMatrix.hpp.

6.150.4.2 `SmartPtr<const MatrixSpace> Ipopt::ScaledMatrixSpace::unscaled_matrix_space_` `[private]`

unscaled matrix space

Definition at line 205 of file IpScaledMatrix.hpp.

6.150.4.3 `SmartPtr<Vector> Ipopt::ScaledMatrixSpace::column_scaling_` `[private]`

column scaling vector

Definition at line 207 of file IpScaledMatrix.hpp.

The documentation for this class was generated from the following file:

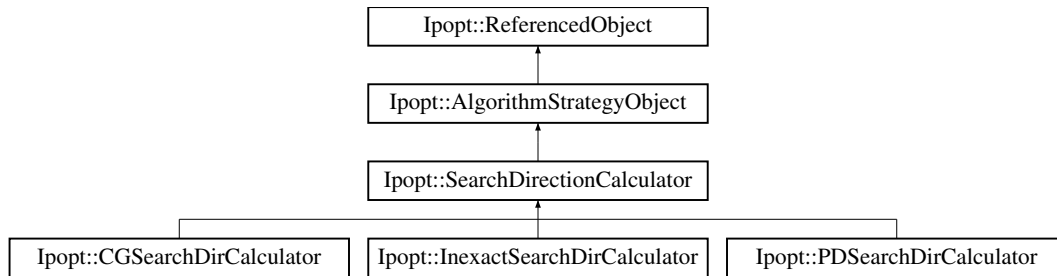
- [LinAlg/IpScaledMatrix.hpp](#)

6.151 Ipopt::SearchDirectionCalculator Class Reference

Base class for computing the search direction for the line search.

```
#include <IpSearchDirCalculator.hpp>
```

Inheritance diagram for Ipopt::SearchDirectionCalculator:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [ComputeSearchDirection](#) ()=0
Pure virtual method for computing the search direction.

Constructors/Destructors

- [SearchDirectionCalculator](#) ()
Constructor.
- virtual [~SearchDirectionCalculator](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [SearchDirectionCalculator](#) (const [SearchDirectionCalculator](#) &)
Default Constructor.
- void [operator=](#) (const [SearchDirectionCalculator](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.151.1 Detailed Description

Base class for computing the search direction for the line search.

Definition at line 20 of file [IpSearchDirCalculator.hpp](#).

6.151.2 Constructor & Destructor Documentation

6.151.2.1 Ipopt::SearchDirectionCalculator::SearchDirectionCalculator () [inline]

Constructor.

Definition at line 26 of file IpSearchDirCalculator.hpp.

6.151.2.2 virtual Ipopt::SearchDirectionCalculator::~~SearchDirectionCalculator () [inline], [virtual]

Default destructor.

Definition at line 30 of file IpSearchDirCalculator.hpp.

6.151.2.3 Ipopt::SearchDirectionCalculator::SearchDirectionCalculator (const SearchDirectionCalculator &) [private]

Default Constructor.

Copy Constructor

6.151.3 Member Function Documentation

6.151.3.1 virtual bool Ipopt::SearchDirectionCalculator::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::InexactSearchDirCalculator](#), [Ipopt::CGSearchDirCalculator](#), and [Ipopt::PDSearchDirCalculator](#).

6.151.3.2 virtual bool Ipopt::SearchDirectionCalculator::ComputeSearchDirection () [pure virtual]

Pure virtual method for computing the search direction.

The computed direction is stored in [IpData\(\).delta\(\)](#).

Implemented in [Ipopt::InexactSearchDirCalculator](#), [Ipopt::CGSearchDirCalculator](#), and [Ipopt::PDSearchDirCalculator](#).

6.151.3.3 void Ipopt::SearchDirectionCalculator::operator= (const SearchDirectionCalculator &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

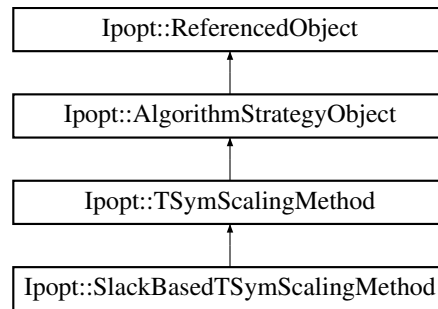
- [Algorithm/IpSearchDirCalculator.hpp](#)

6.152 Ipopt::SlackBasedTSymScalingMethod Class Reference

Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm.

```
#include <IpSlackBasedTSymScalingMethod.hpp>
```

Inheritance diagram for Ipopt::SlackBasedTSymScalingMethod:



Public Member Functions

- virtual bool `InitializeImpl` (const `OptionsList` &options, const std::string &prefix)
overloaded from `AlgorithmStrategyObject`
- virtual bool `ComputeSymTSymScalingFactors` (`Index` n, `Index` nnz, const `ipfint` *airn, const `ipfint` *ajcn, const double *a, double *scaling_factors)
Method for computing the symmetric scaling factors, given the symmetric matrix in triplet (MA27) format.

Constructor/Destructor

- `SlackBasedTSymScalingMethod` ()
- virtual `~SlackBasedTSymScalingMethod` ()

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `SlackBasedTSymScalingMethod` (const `SlackBasedTSymScalingMethod` &)
Copy Constructor.
- void `operator=` (const `SlackBasedTSymScalingMethod` &)
Overloaded Equals Operator.

Additional Inherited Members

6.152.1 Detailed Description

Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm.

The scaling is only considering the current slacks.

Definition at line 23 of file `IpSlackBasedTSymScalingMethod.hpp`.

6.152.2 Constructor & Destructor Documentation

6.152.2.1 `Ipopt::SlackBasedTSymScalingMethod::SlackBasedTSymScalingMethod ()` `[inline]`

Definition at line 28 of file `IpSlackBasedTSymScalingMethod.hpp`.

6.152.2.2 `virtual Ipopt::SlackBasedTSymScalingMethod::~~SlackBasedTSymScalingMethod () [inline],[virtual]`

Definition at line 31 of file `IpSlackBasedTSymScalingMethod.hpp`.

6.152.2.3 `Ipopt::SlackBasedTSymScalingMethod::SlackBasedTSymScalingMethod (const SlackBasedTSymScalingMethod &) [private]`

Copy Constructor.

6.152.3 Member Function Documentation

6.152.3.1 `virtual bool Ipopt::SlackBasedTSymScalingMethod::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::TSymScalingMethod](#).

6.152.3.2 `virtual bool Ipopt::SlackBasedTSymScalingMethod::ComputeSymTScalingFactors (Index n, Index nnz, const ipfint * airn, const ipfint * ajcn, const double * a, double * scaling_factors) [virtual]`

Method for computing the symmetric scaling factors, given the symmetric matrix in triplet (MA27) format.

6.152.3.3 `void Ipopt::SlackBasedTSymScalingMethod::operator= (const SlackBasedTSymScalingMethod &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

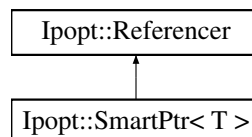
- [Algorithm/LinearSolvers/IpSlackBasedTSymScalingMethod.hpp](#)

6.153 Ipopt::SmartPtr< T > Class Template Reference

Template class for Smart Pointers.

`#include <IpSmartPtr.hpp>`

Inheritance diagram for `Ipopt::SmartPtr< T >`:



Public Member Functions

Constructors/Destructors

- [SmartPtr](#) ()
Default constructor, initialized to NULL.
- [SmartPtr](#) (const [SmartPtr](#)< T > ©)
Copy constructor, initialized from copy of type T.

- `template<class U >`
`SmartPtr (const SmartPtr< U > ©)`
Copy constructor, initialized from copy of type U.
- `SmartPtr (T *ptr)`
Constructor, initialized from T ptr.*
- `~SmartPtr ()`
Destructor, automatically decrements the reference count, deletes the object if necessary.

Friends

friend method declarations.

- `template<class U >`
`U * GetRawPtr (const SmartPtr< U > &smart_ptr)`
Returns the raw pointer contained.
- `template<class U >`
`SmartPtr< const U > ConstPtr (const SmartPtr< U > &smart_ptr)`
Returns a const pointer.
- `template<class U >`
`bool IsValid (const SmartPtr< U > &smart_ptr)`
Returns true if the SmartPtr is NOT NULL.
- `template<class U >`
`bool IsNull (const SmartPtr< U > &smart_ptr)`
Returns true if the SmartPtr is NULL.

Overloaded operators.

- `T * operator-> () const`
Overloaded arrow operator, allows the user to call methods using the contained pointer.
- `T & operator* () const`
Overloaded dereference operator, allows the user to dereference the contained pointer.
- `SmartPtr< T > & operator= (T *rhs)`
Overloaded equals operator, allows the user to set the value of the SmartPtr from a raw pointer.
- `SmartPtr< T > & operator= (const SmartPtr< T > &rhs)`
Overloaded equals operator, allows the user to set the value of the SmartPtr from another SmartPtr.
- `template<class U >`
`SmartPtr< T > & operator= (const SmartPtr< U > &rhs)`
Overloaded equals operator, allows the user to set the value of the SmartPtr from another SmartPtr of a different type.
- `template<class U1 , class U2 >`
`bool operator== (const SmartPtr< U1 > &lhs, const SmartPtr< U2 > &rhs)`
Overloaded equality comparison operator, allows the user to compare the value of two SmartPtrs.
- `template<class U1 , class U2 >`
`bool operator== (const SmartPtr< U1 > &lhs, U2 *raw_rhs)`
Overloaded equality comparison operator, allows the user to compare the value of a SmartPtr with a raw pointer.
- `template<class U1 , class U2 >`
`bool operator== (U1 *lhs, const SmartPtr< U2 > &raw_rhs)`
Overloaded equality comparison operator, allows the user to compare the value of a raw pointer with a SmartPtr.
- `template<class U1 , class U2 >`
`bool operator!= (const SmartPtr< U1 > &lhs, const SmartPtr< U2 > &rhs)`
Overloaded in-equality comparison operator, allows the user to compare the value of two SmartPtrs.

- `template<class U1 , class U2 >`
`bool operator!= (const SmartPtr< U1 > &lhs, U2 *raw_rhs)`
Overloaded in-equality comparison operator, allows the user to compare the value of a [SmartPtr](#) with a raw pointer.
- `template<class U1 , class U2 >`
`bool operator!= (U1 *lhs, const SmartPtr< U2 > &raw_rhs)`
Overloaded in-equality comparison operator, allows the user to compare the value of a [SmartPtr](#) with a raw pointer.
- `template<class U >`
`bool operator< (const SmartPtr< U > &lhs, const SmartPtr< U > &rhs)`
Overloaded less-than comparison operator, allows the user to compare the value of two [SmartPtrs](#).

Private Data/Methods

- `T * ptr_`
Actual raw pointer to the object.
- `SmartPtr< T > & SetFromRawPtr_ (T *rhs)`
Set the value of the internal raw pointer from another raw pointer, releasing the previously referenced object if necessary.
- `SmartPtr< T > & SetFromSmartPtr_ (const SmartPtr< T > &rhs)`
Set the value of the internal raw pointer from a [SmartPtr](#), releasing the previously referenced object if necessary.
- `void ReleasePointer_ ()`
Release the currently referenced object.

6.153.1 Detailed Description

`template<class T>class Ipopt::SmartPtr< T >`

Template class for Smart Pointers.

A [SmartPtr](#) behaves much like a raw pointer, but manages the lifetime of an object, deleting the object automatically. This class implements a reference-counting, intrusive smart pointer design, where all objects pointed to must inherit off of [ReferencedObject](#), which stores the reference count. Although this is intrusive (native types and externally authored classes require wrappers to be referenced by smart pointers), it is a safer design. A more detailed discussion of these issues follows after the usage information.

Usage Example: Note: to use the [SmartPtr](#), all objects to which you point MUST inherit off of [ReferencedObject](#).

```
*
* In MyClass.hpp...
*
* #include "IpReferenced.hpp"

* namespace Ipopt {
*
*   class MyClass : public ReferencedObject // must derive from ReferencedObject
*   {
*       ...
*   }
* } // namespace Ipopt
*
*
* In my_usage.cpp...
*
* #include "IpSmartPtr.hpp"
* #include "MyClass.hpp"
*
* void func(AnyObject& obj)
* {
```

```

*   SmartPtr<MyClass> ptr_to_myclass = new MyClass(...);
*   // ptr_to_myclass now points to a new MyClass,
*   // and the reference count is 1
*
*   ...
*
*   obj.SetMyClass(ptr_to_myclass);
*   // Here, let's assume that AnyObject uses a
*   // SmartPtr<MyClass> internally here.
*   // Now, both ptr_to_myclass and the internal
*   // SmartPtr in obj point to the same MyClass object
*   // and its reference count is 2.
*
*   ...
*
*   // No need to delete ptr_to_myclass, this
*   // will be done automatically when the
*   // reference count drops to zero.
*
* }
*
*

```

It is not necessary to use [SmartPtr](#)'s in all cases where an object is used that has been allocated "into" a [SmartPtr](#). It is possible to just pass objects by reference or regular pointers, even if lower down in the stack a [SmartPtr](#) is to be held on to. Everything should work fine as long as a pointer created by "new" is immediately passed into a [SmartPtr](#), and if [SmartPtr](#)'s are used to hold on to objects.

Other Notes: The [SmartPtr](#) implements both dereference operators -> & *. The [SmartPtr](#) does NOT implement a conversion operator to the raw pointer. Use the [GetRawPtr\(\)](#) method when this is necessary. Make sure that the raw pointer is NOT deleted. The [SmartPtr](#) implements the comparison operators == & != for a variety of types. Use these instead of

```

*   if (GetRawPtr(smrt_ptr) == ptr) // Don't use this
*

```

[SmartPtr](#)'s, as currently implemented, do NOT handle circular references. For example: consider a higher level object using [SmartPtr](#)s to point to A and B, but A and B also point to each other (i.e. A has a [SmartPtr](#) to B and B has a [SmartPtr](#) to A). In this scenario, when the higher level object is finished with A and B, their reference counts will never drop to zero (since they reference each other) and they will not be deleted. This can be detected by memory leak tools like valgrind. If the circular reference is necessary, the problem can be overcome by a number of techniques:

1) A and B can have a method that "releases" each other, that is they set their internal [SmartPtr](#)s to NULL.

```

*   void AClass::ReleaseCircularReferences()
*   {
*       smart_ptr_to_B = NULL;
*   }
*

```

Then, the higher level class can call these methods before it is done using A & B.

2) Raw pointers can be used in A and B to reference each other. Here, an implicit assumption is made that the lifetime is controlled by the higher level object and that A and B will both exist in a controlled manner. Although this seems dangerous, in many situations, this type of referencing is very controlled and this is reasonably safe.

3) This [SmartPtr](#) class could be redesigned with the Weak/Strong design concept. Here, the [SmartPtr](#) is identified as being Strong (controls lifetime of the object) or Weak (merely referencing the object). The Strong [SmartPtr](#) increments (and decrements) the reference count in [ReferencedObject](#) but the Weak [SmartPtr](#) does not. In the example above, the higher level object would have Strong [SmartPtr](#)s to A and B, but A and B would have Weak [SmartPtr](#)s to each other. Then, when the higher level object was done with A and B, they would be deleted. The Weak [SmartPtr](#)s in A and B would not decrement the reference count and would, of course, not delete the object. This idea is very similar to item (2), where it is implied that the sequence of events is controlled such that A and B will not call anything using their

pointers following the higher level delete (i.e. in their destructors!). This is somehow safer, however, because code can be written (however expensive) to perform run-time detection of this situation. For example, the [ReferencedObject](#) could store pointers to all Weak SmartPtrs that are referencing it and, in its destructor, tell these pointers that it is dying. They could then set themselves to NULL, or set an internal flag to detect usage past this point.

Comments on Non-Intrusive Design: In a non-intrusive design, the reference count is stored somewhere other than the object being referenced. This means, unless the reference counting pointer is the first referencer, it must get a pointer to the referenced object from another smart pointer (so it has access to the reference count location). In this non-intrusive design, if we are pointing to an object with a smart pointer (or a number of smart pointers), and we then give another smart pointer the address through a RAW pointer, we will have two independent, AND INCORRECT, reference counts. To avoid this pitfall, we use an intrusive reference counting technique where the reference count is stored in the object being referenced.

Definition at line 172 of file IpSmartPtr.hpp.

6.153.2 Constructor & Destructor Documentation

6.153.2.1 `template<class T> Ipopt::SmartPtr< T >::SmartPtr ()`

Default constructor, initialized to NULL.

Definition at line 361 of file IpSmartPtr.hpp.

6.153.2.2 `template<class T> Ipopt::SmartPtr< T >::SmartPtr (const SmartPtr< T > & copy)`

Copy constructor, initialized from copy of type T.

Definition at line 377 of file IpSmartPtr.hpp.

6.153.2.3 `template<class T> template<class U> Ipopt::SmartPtr< T >::SmartPtr (const SmartPtr< U > & copy)`

Copy constructor, initialized from copy of type U.

Definition at line 395 of file IpSmartPtr.hpp.

6.153.2.4 `template<class T> Ipopt::SmartPtr< T >::SmartPtr (T* ptr)`

Constructor, initialized from T* ptr.

Definition at line 412 of file IpSmartPtr.hpp.

6.153.2.5 `template<class T> Ipopt::SmartPtr< T >::~~SmartPtr ()`

Destructor, automatically decrements the reference count, deletes the object if necessary.

Definition at line 428 of file IpSmartPtr.hpp.

6.153.3 Member Function Documentation

6.153.3.1 `template<class T> T* Ipopt::SmartPtr< T >::operator-> () const`

Overloaded arrow operator, allows the user to call methods using the contained pointer.

Definition at line 439 of file IpSmartPtr.hpp.

6.153.3.2 `template<class T> T& Ipopt::SmartPtr< T >::operator* () const`

Overloaded dereference operator, allows the user to dereference the contained pointer.

Definition at line 455 of file IpSmartPtr.hpp.

6.153.3.3 `template<class T> SmartPtr< T > & Ipopt::SmartPtr< T >::operator= (T * rhs)`

Overloaded equals operator, allows the user to set the value of the [SmartPtr](#) from a raw pointer.

Definition at line 471 of file IpSmartPtr.hpp.

6.153.3.4 `template<class T> SmartPtr< T > & Ipopt::SmartPtr< T >::operator= (const SmartPtr< T > & rhs)`

Overloaded equals operator, allows the user to set the value of the [SmartPtr](#) from another [SmartPtr](#).

Definition at line 482 of file IpSmartPtr.hpp.

6.153.3.5 `template<class T > template<class U > SmartPtr< T > & Ipopt::SmartPtr< T >::operator= (const SmartPtr< U > & rhs)`

Overloaded equals operator, allows the user to set the value of the [SmartPtr](#) from another [SmartPtr](#) of a different type.

Definition at line 496 of file IpSmartPtr.hpp.

6.153.3.6 `template<class T> SmartPtr< T > & Ipopt::SmartPtr< T >::SetFromRawPtr_ (T * rhs) [private]`

Set the value of the internal raw pointer from another raw pointer, releasing the previously referenced object if necessary.

Definition at line 509 of file IpSmartPtr.hpp.

6.153.3.7 `template<class T> SmartPtr< T > & Ipopt::SmartPtr< T >::SetFromSmartPtr_ (const SmartPtr< T > & rhs) [private]`

Set the value of the internal raw pointer from a [SmartPtr](#), releasing the previously referenced object if necessary.

Definition at line 528 of file IpSmartPtr.hpp.

6.153.3.8 `template<class T> void Ipopt::SmartPtr< T >::ReleasePointer_ () [private]`

Release the currently referenced object.

Definition at line 543 of file IpSmartPtr.hpp.

6.153.4 Friends And Related Function Documentation

6.153.4.1 `template<class T> template<class U1 , class U2 > bool operator== (const SmartPtr< U1 > & lhs, const SmartPtr< U2 > & rhs) [friend]`

Overloaded equality comparison operator, allows the user to compare the value of two [SmartPtrs](#).

6.153.4.2 `template<class T> template<class U1 , class U2 > bool operator== (const SmartPtr< U1 > & lhs, U2 * raw_rhs) [friend]`

Overloaded equality comparison operator, allows the user to compare the value of a [SmartPtr](#) with a raw pointer.

6.153.4.3 `template<class T> template<class U1 , class U2 > bool operator== (U1 * lhs, const SmartPtr< U2 > & raw_rhs) [friend]`

Overloaded equality comparison operator, allows the user to compare the value of a raw pointer with a [SmartPtr](#).

6.153.4.4 `template<class T> template<class U1 , class U2 > bool operator!= (const SmartPtr< U1 > & lhs, const SmartPtr< U2 > & rhs) [friend]`

Overloaded in-equality comparison operator, allows the user to compare the value of two SmartPtrs.

6.153.4.5 `template<class T> template<class U1 , class U2 > bool operator!= (const SmartPtr< U1 > & lhs, U2 * raw_rhs) [friend]`

Overloaded in-equality comparison operator, allows the user to compare the value of a SmartPtr with a raw pointer.

6.153.4.6 `template<class T> template<class U1 , class U2 > bool operator!= (U1 * lhs, const SmartPtr< U2 > & raw_rhs) [friend]`

Overloaded in-equality comparison operator, allows the user to compare the value of a SmartPtr with a raw pointer.

6.153.4.7 `template<class T> template<class U > bool operator< (const SmartPtr< U > & lhs, const SmartPtr< U > & rhs) [friend]`

Overloaded less-than comparison operator, allows the user to compare the value of two SmartPtrs.

6.153.4.8 `template<class T> template<class U > U* GetRawPtr (const SmartPtr< U > & smart_ptr) [friend]`

Returns the raw pointer contained.

Use to get the value of the raw ptr (i.e. to pass to other methods/functions, etc.) Note: This method does NOT copy, therefore, modifications using this value modify the underlying object contained by the SmartPtr, NEVER delete this returned value.

6.153.4.9 `template<class T> template<class U > SmartPtr<const U> ConstPtr (const SmartPtr< U > & smart_ptr) [friend]`

Returns a const pointer.

6.153.4.10 `template<class T> template<class U > bool IsValid (const SmartPtr< U > & smart_ptr) [friend]`

Returns true if the SmartPtr is NOT NULL.

Use this to check if the SmartPtr is not null This is preferred to if(GetRawPtr(sp) != NULL)

6.153.4.11 `template<class T> template<class U > bool IsNull (const SmartPtr< U > & smart_ptr) [friend]`

Returns true if the SmartPtr is NULL.

Use this to check if the SmartPtr IsNull. This is preferred to if(GetRawPtr(sp) == NULL)

6.153.5 Member Data Documentation

6.153.5.1 `template<class T> T* Ipopt::SmartPtr< T >::ptr_ [private]`

Actual raw pointer to the object.

Definition at line 308 of file IpSmartPtr.hpp.

The documentation for this class was generated from the following file:

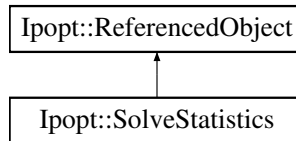
- [Common/IpSmartPtr.hpp](#)

6.154 Ipopt::SolveStatistics Class Reference

This class collects statistics about an optimization run, such as iteration count, final infeasibilities etc.

```
#include <IpSolveStatistics.hpp>
```

Inheritance diagram for Ipopt::SolveStatistics:



Public Member Functions

Constructors/Destructors

- [SolveStatistics](#) (const [SmartPtr](#)< [IpoptNLP](#) > &ip_nlp, const [SmartPtr](#)< [IpoptData](#) > &ip_data, const [SmartPtr](#)< [IpoptCalculatedQuantities](#) > &ip_cq)
Default constructor.
- virtual [~SolveStatistics](#) ()
Default destructor.

Accessor methods for retrieving different kind of solver

statistics information

- virtual [Index](#) [IterationCount](#) () const
Iteration counts.
- virtual [Number](#) [TotalCpuTime](#) () const
Total CPU time, including function evaluations.
- [Number](#) [TotalCPUTime](#) () const
Total CPU time, including function evaluations.
- virtual [Number](#) [TotalSysTime](#) () const
Total System time, including function evaluations.
- virtual [Number](#) [TotalWallclockTime](#) () const
Total wall clock time, including function evaluations.
- virtual void [NumberOfEvaluations](#) ([Index](#) &num_obj_evals, [Index](#) &num_constr_evals, [Index](#) &num_obj_grad_evals, [Index](#) &num_constr_jac_evals, [Index](#) &num_hess_evals) const
Number of NLP function evaluations.
- virtual void [Infeasibilities](#) ([Number](#) &dual_inf, [Number](#) &constr_viol, [Number](#) &complementarity, [Number](#) &kkt_error) const
Unscaled solution infeasibilities.
- virtual void [ScaledInfeasibilities](#) ([Number](#) &scaled_dual_inf, [Number](#) &scaled_constr_viol, [Number](#) &scaled_complementarity, [Number](#) &scaled_kkt_error) const
Scaled solution infeasibilities.
- virtual [Number](#) [FinalObjective](#) () const
Final value of objective function.
- virtual [Number](#) [FinalScaledObjective](#) () const
Final scaled value of objective function.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [SolveStatistics](#) ()
Default Constructor.
- [SolveStatistics](#) (const [SolveStatistics](#) &)
Copy Constructor.
- void [operator=](#) (const [SolveStatistics](#) &)
Overloaded Equals Operator.

Private Attributes

Fields for storing the statistics data

- [Index num_iters_](#)
Number of iterations.
- [Number total_cpu_time_](#)
- [Number total_sys_time_](#)
- [Number total_wallclock_time_](#)
- [Index num_obj_evals_](#)
Number of objective function evaluations.
- [Index num_constr_evals_](#)
Number of constraints evaluations (max of equality and inequality)
- [Index num_obj_grad_evals_](#)
Number of objective gradient evaluations.
- [Index num_constr_jac_evals_](#)
Number of constraint Jacobian evaluations.
- [Index num_hess_evals_](#)
Number of Lagrangian Hessian evaluations.
- [Number scaled_obj_val_](#)
Final scaled value of objective function.
- [Number obj_val_](#)
Final unscaled value of objective function.
- [Number scaled_dual_inf_](#)
Final scaled dual infeasibility (max-norm)
- [Number dual_inf_](#)
Final unscaled dual infeasibility (max-norm)
- [Number scaled_constr_viol_](#)
Final scaled constraint violation (max-norm)
- [Number constr_viol_](#)
Final unscaled constraint violation (max-norm)
- [Number scaled_compl_](#)
Final scaled complementarity error (max-norm)
- [Number compl_](#)
Final unscaled complementarity error (max-norm)
- [Number scaled_kkt_error_](#)
Final overall scaled KKT error (max-norm)
- [Number kkt_error_](#)
Final overall unscaled KKT error (max-norm)

6.154.1 Detailed Description

This class collects statistics about an optimization run, such as iteration count, final infeasibilities etc.

It is meant to provide such information to a user of `Ipopt` during the `finalize_solution` call.

Definition at line 27 of file `IpSolveStatistics.hpp`.

6.154.2 Constructor & Destructor Documentation

6.154.2.1 `Ipopt::SolveStatistics::SolveStatistics (const SmartPtr< IpoptNLP > & ip_nlp, const SmartPtr< IpoptData > & ip_data, const SmartPtr< IpoptCalculatedQuantities > & ip_cq)`

Default constructor.

It takes in those collecting `Ipopt` objects that can provide the statistics information. Those statistics are retrieved at the time of the constructor call.

6.154.2.2 `virtual Ipopt::SolveStatistics::~~SolveStatistics () [inline],[virtual]`

Default destructor.

Definition at line 41 of file `IpSolveStatistics.hpp`.

6.154.2.3 `Ipopt::SolveStatistics::SolveStatistics () [private]`

Default Constructor.

6.154.2.4 `Ipopt::SolveStatistics::SolveStatistics (const SolveStatistics &) [private]`

Copy Constructor.

6.154.3 Member Function Documentation

6.154.3.1 `virtual Index Ipopt::SolveStatistics::IterationCount () const [virtual]`

Iteration counts.

6.154.3.2 `virtual Number Ipopt::SolveStatistics::TotalCpuTime () const [virtual]`

Total CPU time, including function evaluations.

6.154.3.3 `Number Ipopt::SolveStatistics::TotalCPUTime () const [inline]`

Total CPU time, including function evaluations.

Included for backward compatibility.

Definition at line 54 of file `IpSolveStatistics.hpp`.

6.154.3.4 `virtual Number Ipopt::SolveStatistics::TotalSysTime () const [virtual]`

Total System time, including function evaluations.

6.154.3.5 `virtual Number Ipopt::SolveStatistics::TotalWallclockTime () const [virtual]`

Total wall clock time, including function evaluations.

6.154.3.6 `virtual void Ipopt::SolveStatistics::NumberOfEvaluations (Index & num_obj_evals, Index & num_constr_evals, Index & num_obj_grad_evals, Index & num_constr_jac_evals, Index & num_hess_evals) const` [virtual]

Number of [NLP](#) function evaluations.

6.154.3.7 `virtual void Ipopt::SolveStatistics::Infeasibilities (Number & dual_inf, Number & constr_viol, Number & complementarity, Number & kkt_error) const` [virtual]

Unscaled solution infeasibilities.

6.154.3.8 `virtual void Ipopt::SolveStatistics::ScaledInfeasibilities (Number & scaled_dual_inf, Number & scaled_constr_viol, Number & scaled_complementarity, Number & scaled_kkt_error) const` [virtual]

Scaled solution infeasibilities.

6.154.3.9 `virtual Number Ipopt::SolveStatistics::FinalObjective () const` [virtual]

Final value of objective function.

6.154.3.10 `virtual Number Ipopt::SolveStatistics::FinalScaledObjective () const` [virtual]

Final scaled value of objective function.

6.154.3.11 `void Ipopt::SolveStatistics::operator= (const SolveStatistics &)` [private]

Overloaded Equals Operator.

6.154.4 Member Data Documentation

6.154.4.1 `Index Ipopt::SolveStatistics::num_iters_` [private]

Number of iterations.

Definition at line 106 of file `IpSolveStatistics.hpp`.

6.154.4.2 `Number Ipopt::SolveStatistics::total_cpu_time_` [private]

Definition at line 108 of file `IpSolveStatistics.hpp`.

6.154.4.3 `Number Ipopt::SolveStatistics::total_sys_time_` [private]

Definition at line 110 of file `IpSolveStatistics.hpp`.

6.154.4.4 `Number Ipopt::SolveStatistics::total_wallclock_time_` [private]

Definition at line 112 of file `IpSolveStatistics.hpp`.

6.154.4.5 `Index Ipopt::SolveStatistics::num_obj_evals_` [private]

Number of objective function evaluations.

Definition at line 114 of file `IpSolveStatistics.hpp`.

6.154.4.6 `Index Ipopt::SolveStatistics::num_constr_evals_` [private]

Number of constraints evaluations (max of equality and inequality)

Definition at line 117 of file `IpSolveStatistics.hpp`.

6.154.4.7 Index `Ipopt::SolveStatistics::num_obj_grad_evals_` [private]

Number of objective gradient evaluations.

Definition at line 119 of file `IpSolveStatistics.hpp`.

6.154.4.8 Index `Ipopt::SolveStatistics::num_constr_jac_evals_` [private]

Number of constraint Jacobian evaluations.

Definition at line 121 of file `IpSolveStatistics.hpp`.

6.154.4.9 Index `Ipopt::SolveStatistics::num_hess_evals_` [private]

Number of Lagrangian Hessian evaluations.

Definition at line 123 of file `IpSolveStatistics.hpp`.

6.154.4.10 Number `Ipopt::SolveStatistics::scaled_obj_val_` [private]

Final scaled value of objective function.

Definition at line 126 of file `IpSolveStatistics.hpp`.

6.154.4.11 Number `Ipopt::SolveStatistics::obj_val_` [private]

Final unscaled value of objective function.

Definition at line 128 of file `IpSolveStatistics.hpp`.

6.154.4.12 Number `Ipopt::SolveStatistics::scaled_dual_inf_` [private]

Final scaled dual infeasibility (max-norm)

Definition at line 130 of file `IpSolveStatistics.hpp`.

6.154.4.13 Number `Ipopt::SolveStatistics::dual_inf_` [private]

Final unscaled dual infeasibility (max-norm)

Definition at line 132 of file `IpSolveStatistics.hpp`.

6.154.4.14 Number `Ipopt::SolveStatistics::scaled_constr_viol_` [private]

Final scaled constraint violation (max-norm)

Definition at line 134 of file `IpSolveStatistics.hpp`.

6.154.4.15 Number `Ipopt::SolveStatistics::constr_viol_` [private]

Final unscaled constraint violation (max-norm)

Definition at line 136 of file `IpSolveStatistics.hpp`.

6.154.4.16 Number `Ipopt::SolveStatistics::scaled_compl_` [private]

Final scaled complementarity error (max-norm)

Definition at line 138 of file `IpSolveStatistics.hpp`.

6.154.4.17 Number Ipopt::SolveStatistics::compl_ [private]

Final unscaled complementarity error (max-norm)

Definition at line 140 of file IpSolveStatistics.hpp.

6.154.4.18 Number Ipopt::SolveStatistics::scaled_kkt_error_ [private]

Final overall scaled KKT error (max-norm)

Definition at line 142 of file IpSolveStatistics.hpp.

6.154.4.19 Number Ipopt::SolveStatistics::kkt_error_ [private]

Final overall unscaled KKT error (max-norm)

Definition at line 144 of file IpSolveStatistics.hpp.

The documentation for this class was generated from the following file:

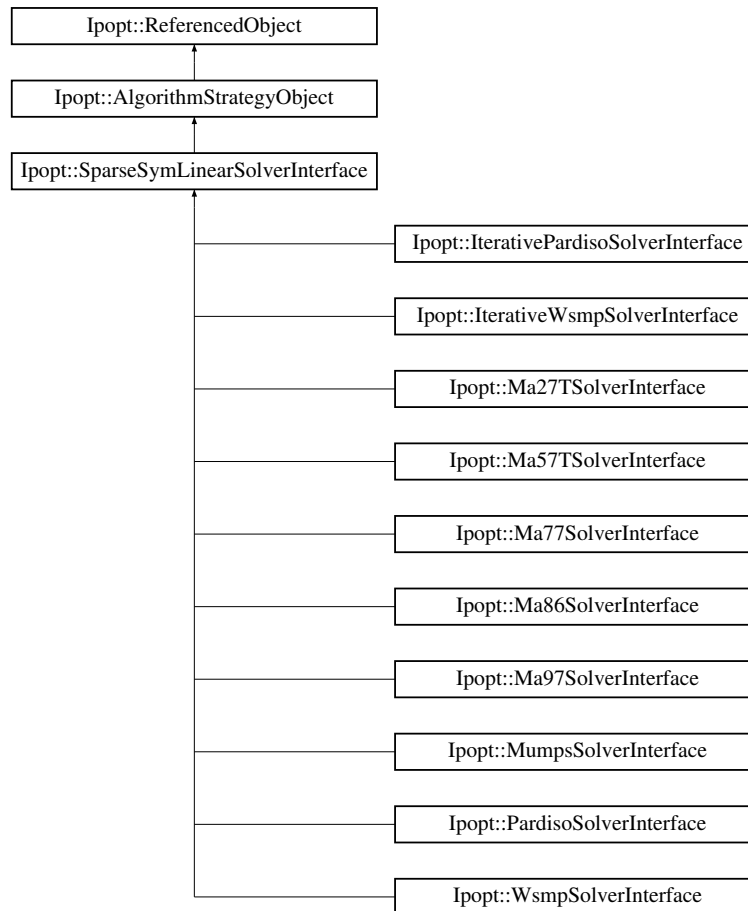
- [Interfaces/IpSolveStatistics.hpp](#)

6.155 Ipopt::SparseSymLinearSolverInterface Class Reference

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

```
#include <IpSparseSymLinearSolverInterface.hpp>
```

Inheritance diagram for Ipopt::SparseSymLinearSolverInterface:



Public Types

- enum [EMatrixFormat](#) {
[Triplet_Format](#), [CSR_Format_0_Offset](#), [CSR_Format_1_Offset](#), [CSR_Full_Format_0_Offset](#),
[CSR_Full_Format_1_Offset](#) }
Enum to specify sparse matrix format.

Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)

Constructor/Destructor

- [SparseSymLinearSolverInterface](#) ()
- virtual [~SparseSymLinearSolverInterface](#) ()

Methods for requesting solution of the linear system.

- virtual [ESymSolverStatus](#) [InitializeStructure](#) ([Index](#) dim, [Index](#) nonzeros, const [Index](#) *ia, const [Index](#) *ja)=0
Method for initializing internal structures.
- virtual double * [GetValuesArrayPtr](#) ()=0

Method returning an internal array into which the nonzero elements (in the same order as ja) will be stored by the calling routine before a call to MultiSolve with a new_matrix=true (or after a return of MultiSolve with SYMSOLV_CALL_AGA-IN).

- virtual [ESymSolverStatus](#) MultiSolve (bool new_matrix, const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals, bool check_NegEVals, [Index](#) numberOfNegEVals)=0

Solve operation for multiple right hand sides.

- virtual [Index](#) NumberOfNegEVals () const =0

Number of negative eigenvalues detected during last factorization.

- virtual bool IncreaseQuality ()=0

Request to increase quality of solution for next solve.

- virtual bool ProvidesInertia () const =0

Query whether inertia is computed by linear solver.

- virtual [EMatrixFormat](#) MatrixFormat () const =0

Query of requested matrix type that the linear solver understands.

Methods related to the detection of linearly dependent

rows in a matrix

- virtual bool ProvidesDegeneracyDetection () const

Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.

- virtual [ESymSolverStatus](#) DetermineDependentRows (const [Index](#) *ia, const [Index](#) *ja, std::list< [Index](#) > &c-_deps)

This method determines the list of row indices of the linearly dependent rows.

Additional Inherited Members

6.155.1 Detailed Description

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

This defines the general interface to linear solvers for sparse symmetric indefinite matrices. The matrices can be provided either in "triplet format" (like for Harwell's MA27 solver), or in compressed sparse row (CSR) format for the lower triangular part of the symmetric matrix.

The solver should be able to compute the inertia of the matrix, or more specifically, the number of negative eigenvalues in the factorized matrix.

This interface is used by the calling objective in the following way:

1. The InitializeImpl method is called at the very beginning (for every optimization run), which allows the linear solver object to retrieve options given in the [OptionsList](#) (such as pivot tolerances etc). At this point, some internal data can also be initialized.
2. The calling class calls MatrixFormat to find out which matrix representation the linear solver requires. The possible options are Triplet_Format, as well as CSR_Format_0_Offset and CSR_Format_1_Offset. The difference between the last two is that for CSR_Format_0_Offset the counting of the element position in the ia and ja arrays starts are 0 (C-style numbering), whereas for the other one it starts at 1 (Fortran-style numbering).
3. After this, the InitializeStructure method is called (once). Here, the structure of the matrix is provided. If the linear solver requires a symbolic preprocessing phase that can be done without knowledge of the matrix element values, it can be done here.

4. The calling class will request an array for storing the actual values for a matrix using the `GetValuesArrayPtr` method. This array must be at least as large as the number of nonzeros in the matrix (as given to this class by the `InitializeStructure` method call). After a call of this method, the calling class will fill this array with the actual values of the matrix.
5. Every time lateron, when actual solves of a linear system is requested, the calling class will call the `MultiSolve` to request the solve, possibly for multiple right-hand sides. The flag `new_matrix` then indicates if the values of the matrix have changed and if a factorization is required, or if an old factorization can be used to do the solve.

Note that the `GetValuesArrayPtr` method will be called before every call of `MultiSolve` with `new_matrix=true`, or before a renewed call of `MultiSolve` if the most previous return value was `SYMSOLV_CALL_AGAIN`.

1. The calling class might request with `NumberOfNegEVals` the number of the negative eigenvalues for the original matrix that were detected during the most recently performed factorization.
2. The calling class might ask the linear solver to increase the quality of the solution. For example, if the linear solver uses a pivot tolerance, a larger value should be used for the next solve (which might require a refactorization).
3. Finally, when the destructor is called, the internal storage, also in the linear solver, should be released.

Note, if the matrix is given in triplet format, entries might be listed multiple times, in which case the corresponding elements have to be added.

A note for warm starts: If the option "warm_start_same_structure" is specified with "yes", the algorithm assumes that a problem with the same sparsity structure is solved for a repeated time. In that case, the linear solver might reuse information from the previous optimization. See [Ma27TSolverInterface](#) for an example.

Definition at line 98 of file `IpSparseSymLinearSolverInterface.hpp`.

6.155.2 Member Enumeration Documentation

6.155.2.1 enum `Ipopt::SparseSymLinearSolverInterface::EMatrixFormat`

Enum to specify sparse matrix format.

Enumerator

- Triplet_Format*** Triplet (MA27) format.
- CSR_Format_0_Offset*** Compressed sparse row format for lower triangular part, with 0 offset.
- CSR_Format_1_Offset*** Compressed sparse row format for lower triangular part, with 1 offset.
- CSR_Full_Format_0_Offset*** Compressed sparse row format for both lwr and upr parts, with 0 offset.
- CSR_Full_Format_1_Offset*** Compressed sparse row format for both lwr and upr parts, with 1 offset.

Definition at line 102 of file `IpSparseSymLinearSolverInterface.hpp`.

6.155.3 Constructor & Destructor Documentation

6.155.3.1 `Ipopt::SparseSymLinearSolverInterface::SparseSymLinearSolverInterface ()` `[inline]`

Definition at line 120 of file `IpSparseSymLinearSolverInterface.hpp`.

6.155.3.2 `virtual Ipopt::SparseSymLinearSolverInterface::~~SparseSymLinearSolverInterface ()` `[inline]`, `[virtual]`

Definition at line 123 of file `IpSparseSymLinearSolverInterface.hpp`.

6.155.4 Member Function Documentation

6.155.4.1 `virtual bool Ipopt::SparseSymLinearSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Ma57-TSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmvSolverInterface](#), [Ipopt::IterativeWsmvSolverInterface](#), and [Ipopt::Ma27TSolverInterface](#).

6.155.4.2 `virtual ESymSolverStatus Ipopt::SparseSymLinearSolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * ia, const Index * ja) [pure virtual]`

Method for initializing internal structures.

Here, ndim gives the number of rows and columns of the matrix, nonzeros give the number of nonzero elements, and ia and ja give the positions of the nonzero elements, given in the matrix format determined by MatrixFormat.

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Ma57-TSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmvSolverInterface](#), [Ipopt::IterativeWsmvSolverInterface](#), and [Ipopt::Ma27TSolverInterface](#).

6.155.4.3 `virtual double* Ipopt::SparseSymLinearSolverInterface::GetValuesArrayPtr () [pure virtual]`

Method returning an internal array into which the nonzero elements (in the same order as ja) will be stored by the calling routine before a call to MultiSolve with a new_matrix=true (or after a return of MultiSolve with SYMSOLV_CALL_AGAIN).

The returned array must have space for at least nonzero elements.

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Ma57-TSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::Ma27TSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmvSolverInterface](#), and [Ipopt::IterativeWsmvSolverInterface](#).

6.155.4.4 `virtual ESymSolverStatus Ipopt::SparseSymLinearSolverInterface::MultiSolve (bool new_matrix, const Index * ia, const Index * ja, Index nrhs, double * rhs_vals, bool check_NegEVals, Index numberOfNegEVals) [pure virtual]`

Solve operation for multiple right hand sides.

Solves the linear system $A * x = b$ with multiple right hand sides, where A is the symmetric indefinite matrix. Here, ia and ja give the positions of the values (in the required matrix data format). The actual values of the matrix will have been given to this object by copying them into the array provided by GetValuesArrayPtr. ia and ja are identical to the ones given to InitializeStructure. The flag new_matrix is set to true, if the values of the matrix has changed, and a refactorization is required.

The return code is SYMSOLV_SUCCESS if the factorization and solves were successful, SYMSOLV_SINGULAR if the linear system is singular, and SYMSOLV_WRONG_INERTIA if check_NegEVals is true and the number of negative eigenvalues in the matrix does not match numberOfNegEVals. If SYMSOLV_CALL_AGAIN is returned, then the calling function will request the pointer for the array for storing a again (with GetValuesPtr), write the values of the nonzero elements into it, and call this MultiSolve method again with the same right-hand sides. (This can be done, for example, if the linear solver realized it does not have sufficient memory and needs to redo the factorization; e.g., for MA27.)

The number of right-hand sides is given by nrhs, the values of the right-hand sides are given in rhs_vals (one full right-hand side stored immediately after the other), and solutions are to be returned in the same array.

check_NegEVals will not be chosen true, if [ProvidesInertia\(\)](#) returns false.

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Ma57-TSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmvSolverInterface](#), and [Ipopt::IterativeWsmvSolverInterface](#).

[TSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::Ma27TSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmSolverInterface](#), and [Ipopt::IterativeWsmSolverInterface](#).

6.155.4.5 `virtual Index Ipopt::SparseSymLinearSolverInterface::NumberOfNegEvals () const [pure virtual]`

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see [ProvidesInertia](#)).

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Ma57-TSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::Ma27TSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmSolverInterface](#), and [Ipopt::IterativeWsmSolverInterface](#).

6.155.4.6 `virtual bool Ipopt::SparseSymLinearSolverInterface::IncreaseQuality () [pure virtual]`

Request to increase quality of solution for next solve.

The calling class asks linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Ma57-TSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::Ma27TSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmSolverInterface](#), and [Ipopt::IterativeWsmSolverInterface](#).

6.155.4.7 `virtual bool Ipopt::SparseSymLinearSolverInterface::ProvidesInertia () const [pure virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Ma57-TSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::Ma27TSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmSolverInterface](#), and [Ipopt::IterativeWsmSolverInterface](#).

6.155.4.8 `virtual EMatrixFormat Ipopt::SparseSymLinearSolverInterface::MatrixFormat () const [pure virtual]`

Query of requested matrix type that the linear solver understands.

Implemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Ma57-TSolverInterface](#), [Ipopt::MumpsSolverInterface](#), [Ipopt::Ma27TSolverInterface](#), [Ipopt::IterativePardisoSolverInterface](#), [Ipopt::PardisoSolverInterface](#), [Ipopt::WsmSolverInterface](#), and [Ipopt::IterativeWsmSolverInterface](#).

6.155.4.9 `virtual bool Ipopt::SparseSymLinearSolverInterface::ProvidesDegeneracyDetection () const [inline], [virtual]`

Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.

Reimplemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Mumps-SolverInterface](#), and [Ipopt::WsmSolverInterface](#).

Definition at line 226 of file [IpSparseSymLinearSolverInterface.hpp](#).

6.155.4.10 `virtual ESymSolverStatus Ipopt::SparseSymLinearSolverInterface::DetermineDependentRows (const Index * ia, const Index * ja, std::list< Index > & c_deps) [inline], [virtual]`

This method determines the list of row indices of the linearly dependent rows.

Reimplemented in [Ipopt::Ma97SolverInterface](#), [Ipopt::Ma86SolverInterface](#), [Ipopt::Ma77SolverInterface](#), [Ipopt::Mumps-SolverInterface](#), and [Ipopt::WsmSolverInterface](#).

Definition at line 232 of file IpSparseSymLinearSolverInterface.hpp.

The documentation for this class was generated from the following file:

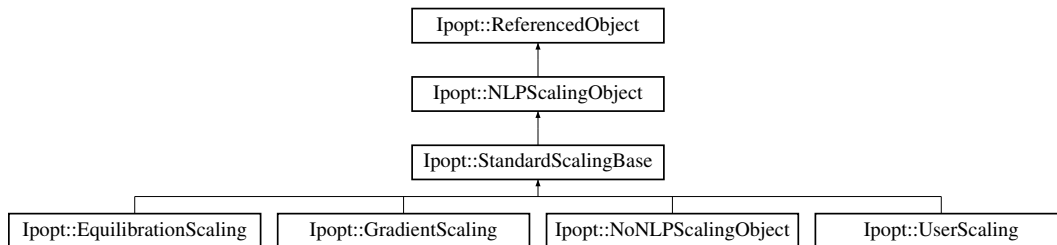
- Algorithm/LinearSolvers/IpSparseSymLinearSolverInterface.hpp

6.156 Ipopt::StandardScalingBase Class Reference

This is a base class for many standard scaling techniques.

```
#include <IpNLPScaling.hpp>
```

Inheritance diagram for Ipopt::StandardScalingBase:



Public Member Functions

- virtual void [DetermineScaling](#) (const [SmartPtr](#)< const [VectorSpace](#) > x_space, const [SmartPtr](#)< const [VectorSpace](#) > c_space, const [SmartPtr](#)< const [VectorSpace](#) > d_space, const [SmartPtr](#)< const [MatrixSpace](#) > jac_c_space, const [SmartPtr](#)< const [MatrixSpace](#) > jac_d_space, const [SmartPtr](#)< const [SymMatrixSpace](#) > h_space, [SmartPtr](#)< const [MatrixSpace](#) > &new_jac_c_space, [SmartPtr](#)< const [MatrixSpace](#) > &new_jac_d_space, [SmartPtr](#)< const [SymMatrixSpace](#) > &new_h_space, const [Matrix](#) &Px_L, const [Vector](#) &x_L, const [Matrix](#) &Px_U, const [Vector](#) &x_U)

This method is called by the [IpoptNLP](#)'s at a convenient time to compute and/or read scaling factors.

Constructors/Destructors

- [StandardScalingBase](#) ()
- virtual [~StandardScalingBase](#) ()
Default destructor.
- virtual [Number](#) [apply_obj_scaling](#) (const [Number](#) &f)
Methods to map scaled and unscaled matrices.
- virtual [Number](#) [unapply_obj_scaling](#) (const [Number](#) &f)
Returns an obj-unscaled version of the given scalar.
- virtual [SmartPtr](#)< [Vector](#) > [apply_vector_scaling_x_NonConst](#) (const [SmartPtr](#)< const [Vector](#) > &v)
Returns an x-scaled version of the given vector.
- virtual [SmartPtr](#)< const [Vector](#) > [apply_vector_scaling_x](#) (const [SmartPtr](#)< const [Vector](#) > &v)
Returns an x-scaled version of the given vector.
- virtual [SmartPtr](#)< [Vector](#) > [unapply_vector_scaling_x_NonConst](#) (const [SmartPtr](#)< const [Vector](#) > &v)
Returns an x-unscaled version of the given vector.
- virtual [SmartPtr](#)< const [Vector](#) > [unapply_vector_scaling_x](#) (const [SmartPtr](#)< const [Vector](#) > &v)
Returns an x-unscaled version of the given vector.

- virtual `SmartPtr< const Vector > apply_vector_scaling_c` (const `SmartPtr< const Vector > &v`)
Returns an c-scaled version of the given vector.
- virtual `SmartPtr< const Vector > unapply_vector_scaling_c` (const `SmartPtr< const Vector > &v`)
Returns an c-unscaled version of the given vector.
- virtual `SmartPtr< Vector > apply_vector_scaling_c_NonConst` (const `SmartPtr< const Vector > &v`)
Returns an c-scaled version of the given vector.
- virtual `SmartPtr< Vector > unapply_vector_scaling_c_NonConst` (const `SmartPtr< const Vector > &v`)
Returns an c-unscaled version of the given vector.
- virtual `SmartPtr< const Vector > apply_vector_scaling_d` (const `SmartPtr< const Vector > &v`)
Returns an d-scaled version of the given vector.
- virtual `SmartPtr< const Vector > unapply_vector_scaling_d` (const `SmartPtr< const Vector > &v`)
Returns an d-unscaled version of the given vector.
- virtual `SmartPtr< Vector > apply_vector_scaling_d_NonConst` (const `SmartPtr< const Vector > &v`)
Returns an d-scaled version of the given vector.
- virtual `SmartPtr< Vector > unapply_vector_scaling_d_NonConst` (const `SmartPtr< const Vector > &v`)
Returns an d-unscaled version of the given vector.
- virtual `SmartPtr< const Matrix > apply_jac_c_scaling` (`SmartPtr< const Matrix > matrix`)
Returns a scaled version of the jacobian for c.
- virtual `SmartPtr< const Matrix > apply_jac_d_scaling` (`SmartPtr< const Matrix > matrix`)
Returns a scaled version of the jacobian for d If the overloaded method does not create a new matrix, make sure to set the matrix ptr passed in to NULL.
- virtual `SmartPtr< const SymMatrix > apply_hessian_scaling` (`SmartPtr< const SymMatrix > matrix`)
Returns a scaled version of the hessian of the lagrangian If the overloaded method does not create a new matrix, make sure to set the matrix ptr passed in to NULL.

Methods for determining whether scaling for entities is

done

- virtual bool `have_x_scaling` ()
Returns true if the primal x variables are scaled.
- virtual bool `have_c_scaling` ()
Returns true if the equality constraints are scaled.
- virtual bool `have_d_scaling` ()
Returns true if the inequality constraints are scaled.

Static Public Member Functions

- static void `RegisterOptions` (`SmartPtr< RegisteredOptions > roptions`)
Methods for IpoptType.

Protected Member Functions

- virtual bool `InitializeImpl` (const `OptionsList &options`, const `std::string &prefix`)
Overloaded initialization method.
- virtual void `DetermineScalingParametersImpl` (const `SmartPtr< const VectorSpace > x_space`, const `SmartPtr< const VectorSpace > c_space`, const `SmartPtr< const VectorSpace > d_space`, const `SmartPtr< const MatrixSpace > jac_c_space`, const `SmartPtr< const MatrixSpace > jac_d_space`, const `SmartPtr< const SymMatrixSpace > h_space`, const `Matrix &Px_L`, const `Vector &x_L`, const `Matrix &Px_U`, const `Vector &x_U`, `Number &df`, `SmartPtr< Vector > &dx`, `SmartPtr< Vector > &dc`, `SmartPtr< Vector > &dd`)=0
This is the method that has to be overloaded by a particular scaling method that somehow computes the scaling vectors dx, dc, and dd.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [StandardScalingBase](#) (const [StandardScalingBase](#) &)
Copy Constructor.
- void `operator=` (const [StandardScalingBase](#) &)
Overloaded Equals Operator.

Private Attributes

- [Number](#) `df_`
Scaling parameters - we only need to keep copies of the objective scaling and the x scaling - the others we can get from the scaled matrix spaces.
- [SmartPtr](#)< [Vector](#) > `dx_`
x scaling
- [SmartPtr](#)< [ScaledMatrixSpace](#) > `scaled_jac_c_space_`
Scaled [Matrix](#) Spaces.
- [SmartPtr](#)< [ScaledMatrixSpace](#) > `scaled_jac_d_space_`
Scaled jacobian of d space.
- [SmartPtr](#)< [SymScaledMatrixSpace](#) > `scaled_h_space_`
Scaled hessian of lagrangian spacea.

Algorithmic parameters

- [Number](#) `obj_scaling_factor_`
Additional scaling value for the objective function.

6.156.1 Detailed Description

This is a base class for many standard scaling techniques.

The overloaded classes only need to provide the scaling parameters

Definition at line 229 of file IpNLPScaling.hpp.

6.156.2 Constructor & Destructor Documentation

6.156.2.1 `Ipopt::StandardScalingBase::StandardScalingBase ()`

6.156.2.2 `virtual Ipopt::StandardScalingBase::~~StandardScalingBase ()` `[virtual]`

Default destructor.

6.156.2.3 `Ipopt::StandardScalingBase::StandardScalingBase (const StandardScalingBase &)` `[private]`

Copy Constructor.

6.156.3 Member Function Documentation

6.156.3.1 **virtual Number** `lpopt::StandardScalingBase::apply_obj_scaling (const Number & f)` `[virtual]`

Methods to map scaled and unscaled matrices.

Returns an obj-scaled version of the given scalar

Implements [lpopt::NLPScalingObject](#).

6.156.3.2 **virtual Number** `lpopt::StandardScalingBase::unapply_obj_scaling (const Number & f)` `[virtual]`

Returns an obj-unscaled version of the given scalar.

Implements [lpopt::NLPScalingObject](#).

6.156.3.3 **virtual SmartPtr<Vector>** `lpopt::StandardScalingBase::apply_vector_scaling_x_NonConst (const SmartPtr< const Vector > & v)` `[virtual]`

Returns an x-scaled version of the given vector.

Implements [lpopt::NLPScalingObject](#).

6.156.3.4 **virtual SmartPtr<const Vector>** `lpopt::StandardScalingBase::apply_vector_scaling_x (const SmartPtr< const Vector > & v)` `[virtual]`

Returns an x-scaled version of the given vector.

Implements [lpopt::NLPScalingObject](#).

6.156.3.5 **virtual SmartPtr<Vector>** `lpopt::StandardScalingBase::unapply_vector_scaling_x_NonConst (const SmartPtr< const Vector > & v)` `[virtual]`

Returns an x-unscaled version of the given vector.

Implements [lpopt::NLPScalingObject](#).

6.156.3.6 **virtual SmartPtr<const Vector>** `lpopt::StandardScalingBase::unapply_vector_scaling_x (const SmartPtr< const Vector > & v)` `[virtual]`

Returns an x-unscaled version of the given vector.

Implements [lpopt::NLPScalingObject](#).

6.156.3.7 **virtual SmartPtr<const Vector>** `lpopt::StandardScalingBase::apply_vector_scaling_c (const SmartPtr< const Vector > & v)` `[virtual]`

Returns an c-scaled version of the given vector.

Implements [lpopt::NLPScalingObject](#).

6.156.3.8 **virtual SmartPtr<const Vector>** `lpopt::StandardScalingBase::unapply_vector_scaling_c (const SmartPtr< const Vector > & v)` `[virtual]`

Returns an c-unscaled version of the given vector.

Implements [lpopt::NLPScalingObject](#).

6.156.3.9 `virtual SmartPtr<Vector> Ipopt::StandardScalingBase::apply_vector_scaling_c_NonConst (const SmartPtr< const Vector > & v) [virtual]`

Returns an c-scaled version of the given vector.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.10 `virtual SmartPtr<Vector> Ipopt::StandardScalingBase::unapply_vector_scaling_c_NonConst (const SmartPtr< const Vector > & v) [virtual]`

Returns an c-unscaled version of the given vector.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.11 `virtual SmartPtr<const Vector> Ipopt::StandardScalingBase::apply_vector_scaling_d (const SmartPtr< const Vector > & v) [virtual]`

Returns an d-scaled version of the given vector.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.12 `virtual SmartPtr<const Vector> Ipopt::StandardScalingBase::unapply_vector_scaling_d (const SmartPtr< const Vector > & v) [virtual]`

Returns an d-unscaled version of the given vector.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.13 `virtual SmartPtr<Vector> Ipopt::StandardScalingBase::apply_vector_scaling_d_NonConst (const SmartPtr< const Vector > & v) [virtual]`

Returns an d-scaled version of the given vector.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.14 `virtual SmartPtr<Vector> Ipopt::StandardScalingBase::unapply_vector_scaling_d_NonConst (const SmartPtr< const Vector > & v) [virtual]`

Returns an d-unscaled version of the given vector.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.15 `virtual SmartPtr<const Matrix> Ipopt::StandardScalingBase::apply_jac_c_scaling (SmartPtr< const Matrix > matrix) [virtual]`

Returns a scaled version of the jacobian for c.

If the overloaded method does not make a new matrix, make sure to set the matrix ptr passed in to NULL.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.16 `virtual SmartPtr<const Matrix> Ipopt::StandardScalingBase::apply_jac_d_scaling (SmartPtr< const Matrix > matrix) [virtual]`

Returns a scaled version of the jacobian for d If the overloaded method does not create a new matrix, make sure to set the matrix ptr passed in to NULL.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.17 `virtual SmartPtr<const SymMatrix> Ipopt::StandardScalingBase::apply_hessian_scaling (SmartPtr< const SymMatrix > matrix) [virtual]`

Returns a scaled version of the hessian of the lagrangian If the overloaded method does not create a new matrix, make sure to set the matrix ptr passed in to NULL.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.18 `virtual bool Ipopt::StandardScalingBase::have_x_scaling () [virtual]`

Returns true if the primal x variables are scaled.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.19 `virtual bool Ipopt::StandardScalingBase::have_c_scaling () [virtual]`

Returns true if the equality constraints are scaled.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.20 `virtual bool Ipopt::StandardScalingBase::have_d_scaling () [virtual]`

Returns true if the inequality constraints are scaled.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.21 `virtual void Ipopt::StandardScalingBase::DetermineScaling (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, SmartPtr< const MatrixSpace > & new_jac_c_space, SmartPtr< const MatrixSpace > & new_jac_d_space, SmartPtr< const SymMatrixSpace > & new_h_space, const Matrix & Px_L, const Vector & x_L, const Matrix & Px_U, const Vector & x_U) [virtual]`

This method is called by the [IpoptNLP](#)'s at a convenient time to compute and/or read scaling factors.

Implements [Ipopt::NLPScalingObject](#).

6.156.3.22 `static void Ipopt::StandardScalingBase::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for [IpoptType](#).

6.156.3.23 `virtual bool Ipopt::StandardScalingBase::InitializeImpl (const OptionsList & options, const std::string & prefix) [protected], [virtual]`

Overloaded initialization method.

Implements [Ipopt::NLPScalingObject](#).

Reimplemented in [Ipopt::EquilibrationScaling](#), and [Ipopt::GradientScaling](#).

6.156.3.24 `virtual void Ipopt::StandardScalingBase::DetermineScalingParametersImpl (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, const Matrix & Px_L, const Vector & x_L, const Matrix & Px_U, const Vector & x_U, Number & df, SmartPtr< Vector > & dx, SmartPtr< Vector > & dc, SmartPtr< Vector > & dd) [protected], [pure virtual]`

This is the method that has to be overloaded by a particular scaling method that somehow computes the scaling vectors dx, dc, and dd.

The pointers to those vectors can be NULL, in which case no scaling for that item will be done later.

Implemented in [Ipopt::NoNLPScalingObject](#), [Ipopt::EquilibrationScaling](#), [Ipopt::GradientScaling](#), and [Ipopt::UserScaling](#).

6.156.3.25 `void Ipopt::StandardScalingBase::operator=(const StandardScalingBase &) [private]`

Overloaded Equals Operator.

6.156.4 Member Data Documentation

6.156.4.1 **Number** `Ipopt::StandardScalingBase::df_ [private]`

Scaling parameters - we only need to keep copies of the objective scaling and the x scaling - the others we can get from the scaled matrix spaces.

objective scaling parameter

Definition at line 377 of file [IpNLPScaling.hpp](#).

6.156.4.2 **SmartPtr<Vector>** `Ipopt::StandardScalingBase::dx_ [private]`

x scaling

Definition at line 379 of file [IpNLPScaling.hpp](#).

6.156.4.3 **SmartPtr<ScaledMatrixSpace>** `Ipopt::StandardScalingBase::scaled_jac_c_space_ [private]`

Scaled [Matrix](#) Spaces.

Scaled jacobian of c space

Definition at line 385 of file [IpNLPScaling.hpp](#).

6.156.4.4 **SmartPtr<ScaledMatrixSpace>** `Ipopt::StandardScalingBase::scaled_jac_d_space_ [private]`

Scaled jacobian of d space.

Definition at line 387 of file [IpNLPScaling.hpp](#).

6.156.4.5 **SmartPtr<SymScaledMatrixSpace>** `Ipopt::StandardScalingBase::scaled_h_space_ [private]`

Scaled hessian of lagrangian spacea.

Definition at line 389 of file [IpNLPScaling.hpp](#).

6.156.4.6 **Number** `Ipopt::StandardScalingBase::obj_scaling_factor_ [private]`

Additional scaling value for the objective function.

Definition at line 395 of file [IpNLPScaling.hpp](#).

The documentation for this class was generated from the following file:

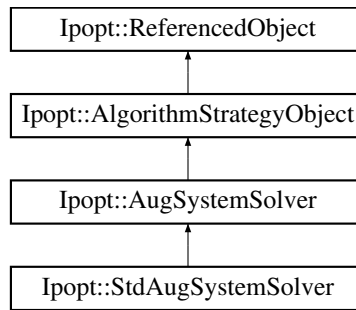
- [Algorithm/IpNLPScaling.hpp](#)

6.157 Ipopt::StdAugSystemSolver Class Reference

Solver for the augmented system for triple type matrices.

```
#include <IpStdAugSystemSolver.hpp>
```

Inheritance diagram for Ipopt::StdAugSystemSolver:



Public Member Functions

- bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual [ESymSolverStatus](#) [MultiSolve](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) *J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) *J_d, const [Vector](#) *D_d, double delta_d, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_xV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_sV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_cV, std::vector< [SmartPtr](#)< const [Vector](#) > > &rhs_dV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_xV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_sV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_cV, std::vector< [SmartPtr](#)< [Vector](#) > > &sol_dV, bool check_NegEVals, [Index](#) numberOfNegEVals)
Set up the augmented system and solve it for a set of given right hand side - implementation for [GenTMatrices](#) and [SymTMatrices](#).
- virtual [Index](#) [NumberOfNegEVals](#) () const
Number of negative eigenvalues detected during last solve.
- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- virtual bool [IncreaseQuality](#) ()
Request to increase quality of solution for next solve.

Constructors/Destructors

- [StdAugSystemSolver](#) ([SymLinearSolver](#) &LinSolver)
Constructor using only a linear solver object.
- virtual [~StdAugSystemSolver](#) ()
Default destructor.

Private Member Functions

- void [CreateAugmentedSpace](#) (const [SymMatrix](#) &W, const [Matrix](#) &J_c, const [Matrix](#) &J_d, const [Vector](#) &proto_x, const [Vector](#) &proto_s, const [Vector](#) &proto_c, const [Vector](#) &proto_d)
Create the matrix space for the Compound Sym [Matrix](#) that represents the augmented system.
- void [CreateAugmentedSystem](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d, const [Vector](#) &proto_x, const [Vector](#) &proto_s, const [Vector](#) &proto_c, const [Vector](#) &proto_d)
Create the new compound sym matrix that represents the augmented system.

- bool [AugmentedSystemRequiresChange](#) (const [SymMatrix](#) *W, double W_factor, const [Vector](#) *D_x, double delta_x, const [Vector](#) *D_s, double delta_s, const [Matrix](#) &J_c, const [Vector](#) *D_c, double delta_c, const [Matrix](#) &J_d, const [Vector](#) *D_d, double delta_d)

Check the internal tags and decide if the passed variables are different from what is in the augmented_system_.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [StdAugSystemSolver](#) ()
Default constructor.
- [StdAugSystemSolver](#) (const [StdAugSystemSolver](#) &)
Copy Constructor.
- void [operator=](#) (const [StdAugSystemSolver](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [SymLinearSolver](#) > [linsolver_](#)
The linear solver object that is to be used to solve the linear systems.
- [SmartPtr](#)< [CompoundSymMatrixSpace](#) > [augmented_system_space_](#)
Spaces for piecing together the augmented system.
- [SmartPtr](#)< [SumSymMatrixSpace](#) > [sumsym_space_x_](#)
- [SmartPtr](#)< [DiagMatrixSpace](#) > [diag_space_x_](#)
- [SmartPtr](#)< [DiagMatrixSpace](#) > [diag_space_s_](#)
- [SmartPtr](#)< [DiagMatrixSpace](#) > [diag_space_c_](#)
- [SmartPtr](#)< [IdentityMatrixSpace](#) > [ident_space_ds_](#)
- [SmartPtr](#)< [DiagMatrixSpace](#) > [diag_space_d_](#)
- [SmartPtr](#)< [CompoundVectorSpace](#) > [augmented_vector_space_](#)
- [SmartPtr](#)< [CompoundSymMatrix](#) > [augmented_system_](#)
The resulting augmented matrix.
- [SmartPtr](#)< const [SymMatrix](#) > [old_w_](#)
A copy of a previous W used in the augmented_system_.

Tags and values to track in order to decide whether the

matrix has to be updated compared to the most recent call of the Set method.

- [TaggedObject::Tag](#) [w_tag_](#)
Tag for W matrix.
- double [w_factor_](#)
Most recent value of W_factor.
- [TaggedObject::Tag](#) [d_x_tag_](#)
Tag for D_x vector, representing the diagonal matrix D_x.
- double [delta_x_](#)
Most recent value of delta_x from Set method.
- [TaggedObject::Tag](#) [d_s_tag_](#)
Tag for D_s vector, representing the diagonal matrix D_s.
- double [delta_s_](#)
Most recent value of delta_s from Set method.
- [TaggedObject::Tag](#) [j_c_tag_](#)

- *Tag for J_c matrix.*
[TaggedObject::Tag d_c_tag_](#)
- *Tag for D_c vector, representing the diagonal matrix D_c .*
 double [delta_c_](#)
- *Most recent value of δ_c from Set method.*
[TaggedObject::Tag j_d_tag_](#)
- *Tag for J_d matrix.*
[TaggedObject::Tag d_d_tag_](#)
- *Tag for D_d vector, representing the diagonal matrix D_d .*
 double [delta_d_](#)
- *Most recent value of δ_d from Set method.*
[TaggedObject::Tag augsys_tag_](#)
- *This is the tag of the matrix storing the augmented system.*

Algorithmic parameters

- bool [warm_start_same_structure_](#)
Flag indicating whether the $TNLP$ with identical structure has already been solved before.

Additional Inherited Members

6.157.1 Detailed Description

Solver for the augmented system for triple type matrices.

The current implemetation assumes that all matrices are of the type [SymTMatrix](#), and all vectors are of the type [Dense-Vector](#).

Definition at line 27 of file `IpStdAugSystemSolver.hpp`.

6.157.2 Constructor & Destructor Documentation

6.157.2.1 `Ipopt::StdAugSystemSolver::StdAugSystemSolver (SymLinearSolver & LinSolver)`

Constructor using only a linear solver object.

6.157.2.2 `virtual Ipopt::StdAugSystemSolver::~~StdAugSystemSolver () [virtual]`

Default destructor.

6.157.2.3 `Ipopt::StdAugSystemSolver::StdAugSystemSolver () [private]`

Default constructor.

6.157.2.4 `Ipopt::StdAugSystemSolver::StdAugSystemSolver (const StdAugSystemSolver &) [private]`

Copy Constructor.

6.157.3 Member Function Documentation

6.157.3.1 `bool Ipopt::StdAugSystemSolver::InitializImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AugSystemSolver](#).

6.157.3.2 `virtual ESymSolverStatus Ipopt::StdAugSystemSolver::MultiSolve (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix * J_c, const Vector * D_c, double delta_c, const Matrix * J_d, const Vector * D_d, double delta_d, std::vector< SmartPtr< const Vector > > & rhs_xV, std::vector< SmartPtr< const Vector > > & rhs_sV, std::vector< SmartPtr< const Vector > > & rhs_cV, std::vector< SmartPtr< const Vector > > & rhs_dV, std::vector< SmartPtr< Vector > > & sol_xV, std::vector< SmartPtr< Vector > > & sol_sV, std::vector< SmartPtr< Vector > > & sol_cV, std::vector< SmartPtr< Vector > > & sol_dV, bool check_NegEVals, Index numberOfNegEVals) [virtual]`

Set up the augmented system and solve it for a set of given right hand side - implementation for GenTMatrices and SymTMatrices.

Reimplemented from [Ipopt::AugSystemSolver](#).

6.157.3.3 `virtual Index Ipopt::StdAugSystemSolver::NumberOfNegEVals () const [virtual]`

Number of negative eigenvalues detected during last solve.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see ProvidesInertia).

Implements [Ipopt::AugSystemSolver](#).

6.157.3.4 `virtual bool Ipopt::StdAugSystemSolver::ProvidesInertia () const [virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::AugSystemSolver](#).

6.157.3.5 `virtual bool Ipopt::StdAugSystemSolver::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

Ask underlying linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::AugSystemSolver](#).

6.157.3.6 `void Ipopt::StdAugSystemSolver::operator= (const StdAugSystemSolver &) [private]`

Overloaded Equals Operator.

6.157.3.7 `void Ipopt::StdAugSystemSolver::CreateAugmentedSpace (const SymMatrix & W, const Matrix & J_c, const Matrix & J_d, const Vector & proto_x, const Vector & proto_s, const Vector & proto_c, const Vector & proto_d) [private]`

Create the matrix space for the Compound Sym [Matrix](#) that represents the augmented system.

This signifies the "first" time through and requires all structural knowledge

6.157.3.8 `void Ipopt::StdAugSystemSolver::CreateAugmentedSystem (const SymMatrix * W, double W_factor, const Vector * D_x, double delta_x, const Vector * D_s, double delta_s, const Matrix & J_c, const Vector * D_c, double delta_c, const Matrix & J_d, const Vector * D_d, double delta_d, const Vector & proto_x, const Vector & proto_s, const Vector & proto_c, const Vector & proto_d) [private]`

Create the new compound sym matrix that represents the augmented system.

This is done EVERY time Solve is called with ANY different information

6.157.3.9 **bool** Ipopt::StdAugSystemSolver::AugmentedSystemRequiresChange (*const SymMatrix * W*, *double W_factor*, *const Vector * D_x*, *double delta_x*, *const Vector * D_s*, *double delta_s*, *const Matrix & J_c*, *const Vector * D_c*, *double delta_c*, *const Matrix & J_d*, *const Vector * D_d*, *double delta_d*) [private]

Check the internal tags and decide if the passed variables are different from what is in the augmented_system_.

6.157.4 Member Data Documentation

6.157.4.1 **SmartPtr<SymLinearSolver>** Ipopt::StdAugSystemSolver::linsolver_ [private]

The linear solver object that is to be used to solve the linear systems.

Definition at line 161 of file IpStdAugSystemSolver.hpp.

6.157.4.2 **SmartPtr<CompoundSymMatrixSpace>** Ipopt::StdAugSystemSolver::augmented_system_space_ [private]

Spaces for piecing together the augmented system.

Definition at line 164 of file IpStdAugSystemSolver.hpp.

6.157.4.3 **SmartPtr<SumSymMatrixSpace>** Ipopt::StdAugSystemSolver::sumsym_space_x_ [private]

Definition at line 165 of file IpStdAugSystemSolver.hpp.

6.157.4.4 **SmartPtr<DiagMatrixSpace>** Ipopt::StdAugSystemSolver::diag_space_x_ [private]

Definition at line 166 of file IpStdAugSystemSolver.hpp.

6.157.4.5 **SmartPtr<DiagMatrixSpace>** Ipopt::StdAugSystemSolver::diag_space_s_ [private]

Definition at line 167 of file IpStdAugSystemSolver.hpp.

6.157.4.6 **SmartPtr<DiagMatrixSpace>** Ipopt::StdAugSystemSolver::diag_space_c_ [private]

Definition at line 168 of file IpStdAugSystemSolver.hpp.

6.157.4.7 **SmartPtr<IdentityMatrixSpace>** Ipopt::StdAugSystemSolver::ident_space_ds_ [private]

Definition at line 169 of file IpStdAugSystemSolver.hpp.

6.157.4.8 **SmartPtr<DiagMatrixSpace>** Ipopt::StdAugSystemSolver::diag_space_d_ [private]

Definition at line 170 of file IpStdAugSystemSolver.hpp.

6.157.4.9 **SmartPtr<CompoundVectorSpace>** Ipopt::StdAugSystemSolver::augmented_vector_space_ [private]

Definition at line 172 of file IpStdAugSystemSolver.hpp.

6.157.4.10 **TaggedObject::Tag** Ipopt::StdAugSystemSolver::w_tag_ [private]

Tag for W matrix.

If W has been given to Set as NULL, then this tag is set to 0

Definition at line 182 of file IpStdAugSystemSolver.hpp.

6.157.4.11 `double Ipopt::StdAugSystemSolver::w_factor_ [private]`

Most recent value of W_factor.

Definition at line 184 of file IpStdAugSystemSolver.hpp.

6.157.4.12 `TaggedObject::Tag Ipopt::StdAugSystemSolver::d_x_tag_ [private]`

Tag for D_x vector, representing the diagonal matrix D_x.

If D_x has been given to Set as NULL, then this tag is set to 0

Definition at line 188 of file IpStdAugSystemSolver.hpp.

6.157.4.13 `double Ipopt::StdAugSystemSolver::delta_x_ [private]`

Most recent value of delta_x from Set method.

Definition at line 190 of file IpStdAugSystemSolver.hpp.

6.157.4.14 `TaggedObject::Tag Ipopt::StdAugSystemSolver::d_s_tag_ [private]`

Tag for D_s vector, representing the diagonal matrix D_s.

If D_s has been given to Set as NULL, then this tag is set to 0

Definition at line 194 of file IpStdAugSystemSolver.hpp.

6.157.4.15 `double Ipopt::StdAugSystemSolver::delta_s_ [private]`

Most recent value of delta_s from Set method.

Definition at line 196 of file IpStdAugSystemSolver.hpp.

6.157.4.16 `TaggedObject::Tag Ipopt::StdAugSystemSolver::j_c_tag_ [private]`

Tag for J_c matrix.

If J_c has been given to Set as NULL, then this tag is set to 0

Definition at line 200 of file IpStdAugSystemSolver.hpp.

6.157.4.17 `TaggedObject::Tag Ipopt::StdAugSystemSolver::d_c_tag_ [private]`

Tag for D_c vector, representing the diagonal matrix D_c.

If D_c has been given to Set as NULL, then this tag is set to 0

Definition at line 204 of file IpStdAugSystemSolver.hpp.

6.157.4.18 `double Ipopt::StdAugSystemSolver::delta_c_ [private]`

Most recent value of delta_c from Set method.

Definition at line 206 of file IpStdAugSystemSolver.hpp.

6.157.4.19 `TaggedObject::Tag Ipopt::StdAugSystemSolver::j_d_tag_ [private]`

Tag for J_d matrix.

If J_d has been given to Set as NULL, then this tag is set to 0

Definition at line 210 of file IpStdAugSystemSolver.hpp.

6.157.4.20 **TaggedObject::Tag** `Ipopt::StdAugSystemSolver::d_d_tag_` [private]

Tag for D_d vector, representing the diagonal matrix D_d.

If D_d has been given to Set as NULL, then this tag is set to 0

Definition at line 214 of file `IpStdAugSystemSolver.hpp`.

6.157.4.21 **double** `Ipopt::StdAugSystemSolver::delta_d_` [private]

Most recent value of delta_d from Set method.

Definition at line 216 of file `IpStdAugSystemSolver.hpp`.

6.157.4.22 **TaggedObject::Tag** `Ipopt::StdAugSystemSolver::augsys_tag_` [private]

This is the tag of the matrix storing the augmented system.

Since this object owns this matrix, no changes should happen outside. However, since it is given away as a smart pointer, someone outside might change it. For debugging purposes, we now track its tag as well.

Definition at line 224 of file `IpStdAugSystemSolver.hpp`.

6.157.4.23 **SmartPtr<CompoundSymMatrix>** `Ipopt::StdAugSystemSolver::augmented_system_` [private]

The resulting augmented matrix.

This matrix is stored as follows: First we have the diagonal elements for the upper left block (for D_W and delta_W), then the elements for the Hessian W, then the Jacobian A, and finally the diagonal elements for the lower right block (for D_C and delta_C).

Definition at line 233 of file `IpStdAugSystemSolver.hpp`.

6.157.4.24 **SmartPtr<const SymMatrix>** `Ipopt::StdAugSystemSolver::old_w_` [private]

A copy of a previous W used in the augmented_system_.

Since Solve can be called with a NULL W, we keep a copy of the last W passed to keep the nonzero structure of the augmented_system_ consistent

Definition at line 238 of file `IpStdAugSystemSolver.hpp`.

6.157.4.25 **bool** `Ipopt::StdAugSystemSolver::warm_start_same_structure_` [private]

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 244 of file `IpStdAugSystemSolver.hpp`.

The documentation for this class was generated from the following file:

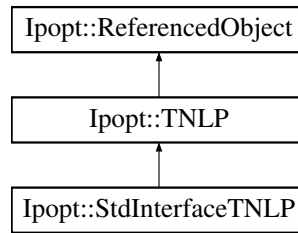
- [Algorithm/IpStdAugSystemSolver.hpp](#)

6.158 **Ipopt::StdInterfaceTNLP Class Reference**

Implementation of a [TNLP](#) for the Standard C interface.

```
#include <IpStdInterfaceTNLP.hpp>
```

Inheritance diagram for `Ipopt::StdInterfaceTNLP`:



Public Member Functions

Constructors/Destructors

- **StdInterfaceTNLP** (Index n_var, const Number *x_L, const Number *x_U, Index n_con, const Number *g_L, const Number *g_U, Index nele_jac, Index nele_hess, Index index_style, const Number *start_x, const Number *start_lam, const Number *start_z_L, const Number *start_z_U, Eval_F_CB eval_f, Eval_G_CB eval_g, Eval_Grad_F_CB eval_grad_f, Eval_Jac_G_CB eval_jac_g, Eval_H_CB eval_h, Intermediate_CB intermediate_cb, Number *x_sol, Number *z_L_sol, Number *z_U_sol, Number *g_sol, Number *lam_sol, Number *obj_sol, UserDataPtr user_data, Number obj_scaling=1, const Number *x_scaling=NULL, const Number *g_scaling=NULL)

Constructor, given dimensions of problem, function pointers for evaluation callback functions, and starting points.

- virtual **~StdInterfaceTNLP** ()

Default destructor.

methods to gather information about the NLP. These methods are

overloaded from **TNLP**.

See **TNLP** for their more detailed documentation.

- virtual bool **get_nlp_info** (Index &n, Index &m, Index &nnz_jac_g, Index &nnz_h_lag, IndexStyleEnum &index_style)
returns dimensions of the nlp.
- virtual bool **get_bounds_info** (Index n, Number *x_l, Number *x_u, Index m, Number *g_l, Number *g_u)
returns bounds of the nlp.
- virtual bool **get_scaling_parameters** (Number &obj_scaling, bool &use_x_scaling, Index n, Number *x_scaling, bool &use_g_scaling, Index m, Number *g_scaling)
returns scaling parameters (if nlp_scaling_method is selected as user-scaling).
- virtual bool **get_starting_point** (Index n, bool init_x, Number *x, bool init_z, Number *z_L, Number *z_U, Index m, bool init_lambda, Number *lambda)
provides a starting point for the nlp variables.
- virtual bool **eval_f** (Index n, const Number *x, bool new_x, Number &obj_value)
evaluates the objective value for the nlp.
- virtual bool **eval_grad_f** (Index n, const Number *x, bool new_x, Number *grad_f)
evaluates the gradient of the objective for the nlp.
- virtual bool **eval_g** (Index n, const Number *x, bool new_x, Index m, Number *g)
evaluates the constraint residuals for the nlp.
- virtual bool **eval_jac_g** (Index n, const Number *x, bool new_x, Index m, Index nele_jac, Index *iRow, Index *jCol, Number *values)
specifies the jacobian structure (if values is NULL) and evaluates the jacobian values (if values is not NULL) for the nlp.
- virtual bool **eval_h** (Index n, const Number *x, bool new_x, Number obj_factor, Index m, const Number *lambda, bool new_lambda, Index nele_hess, Index *iRow, Index *jCol, Number *values)
specifies the structure of the hessian of the lagrangian (if values is NULL) and evaluates the values (if values is not NULL).

- virtual bool `intermediate_callback` (`AlgorithmMode` mode, `Index` iter, `Number` obj_value, `Number` inf_pr, `Number` inf_du, `Number` mu, `Number` d_norm, `Number` regularization_size, `Number` alpha_du, `Number` alpha_pr, `Index` ls_trials, const `IpoptData` *ip_data, `IpoptCalculatedQuantities` *ip_cq)

Intermediate Callback method for the user.

Solution Methods

- virtual void `finalize_solution` (`SolverReturn` status, `Index` n, const `Number` *x, const `Number` *z_L, const `Number` *z_U, `Index` m, const `Number` *g, const `Number` *lambda, `Number` obj_value, const `IpoptData` *ip_data, `IpoptCalculatedQuantities` *ip_cq)

This method is called when the algorithm is complete so the `TNLP` can store/write the solution.

Private Member Functions

- void `apply_new_x` (bool new_x, `Index` n, const `Number` *x)

Internal function to update the internal and ampl state if the x value changes.

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `StdInterfaceTNLP` ()
Default Constructor.
- `StdInterfaceTNLP` (const `StdInterfaceTNLP` &)
Copy Constructor.
- void `operator=` (const `StdInterfaceTNLP` &)
Overloaded Equals Operator.

Private Attributes

- `SmartPtr`< const `Journalist` > `jnlst_`
Journalist.
- `Number` * `non_const_x_`
A non-const copy of x - this is kept up-to-date in `apply_new_x`.
- `Number` * `x_sol_`
Pointers to the user provided vectors for solution.
- `Number` * `z_L_sol_`
- `Number` * `z_U_sol_`
- `Number` * `g_sol_`
- `Number` * `lambda_sol_`
- `Number` * `obj_sol_`

Information about the problem

- const `Index` `n_var_`
Number of variables.
- const `Index` `n_con_`
Number of constraints.
- const `Number` * `x_L_`
Pointer to Number array containing lower bounds for variables.

- const [Number](#) * [x_U_](#)
Pointer to Number array containing upper bounds for variables.
- const [Number](#) * [g_L_](#)
Pointer to Number array containing lower bounds for constraints.
- const [Number](#) * [g_U_](#)
Pointer to Number array containing upper bounds for constraints.
- const [Index](#) [nele_jac_](#)
Number of non-zero elements in the constraint Jacobian.
- const [Index](#) [nele_hess_](#)
Number of non-zero elements in the Hessian.
- const [Index](#) [index_style_](#)
Starting value of the iRow and jCol parameters for matrices.
- const [Number](#) * [start_x_](#)
Pointer to Number array containing starting point for variables.
- const [Number](#) * [start_lam_](#)
Pointer to Number array containing starting values for constraint multipliers.
- const [Number](#) * [start_z_L_](#)
Pointer to Number array containing starting values for lower bound multipliers.
- const [Number](#) * [start_z_U_](#)
Pointer to Number array containing starting values for upper bound multipliers.
- [Eval_F_CB](#) [eval_f_](#)
Pointer to callback function evaluating value of objective function.
- [Eval_G_CB](#) [eval_g_](#)
Pointer to callback function evaluating value of constraints.
- [Eval_Grad_F_CB](#) [eval_grad_f_](#)
Pointer to callback function evaluating gradient of objective function.
- [Eval_Jac_G_CB](#) [eval_jac_g_](#)
Pointer to callback function evaluating Jacobian of constraints.
- [Eval_H_CB](#) [eval_h_](#)
Pointer to callback function evaluating Hessian of Lagrangian.
- [Intermediate_CB](#) [intermediate_cb_](#)
Pointer to intermediate callback function giving control to user.
- [UserDataPtr](#) [user_data_](#)
Pointer to user data.
- [Number](#) [obj_scaling_](#)
Objective scaling factor.
- const [Number](#) * [x_scaling_](#)
Scaling factors for variables (if not NULL)
- const [Number](#) * [g_scaling_](#)
Scaling factors for constraints (if not NULL)

Additional Inherited Members

6.158.1 Detailed Description

Implementation of a [TNLP](#) for the Standard C interface.

The standard C interface is exposed to the user as a single C function that is given problem dimension, starting points, and pointers for functions that evaluate objective function etc.

Definition at line 30 of file `IpStdInterfaceTNLP.hpp`.

6.158.2 Constructor & Destructor Documentation

6.158.2.1 `Ipopt::StdInterfaceTNLP::StdInterfaceTNLP (Index n_var, const Number * x_L, const Number * x_U, Index n_con, const Number * g_L, const Number * g_U, Index nele_jac, Index nele_hess, Index index_style, const Number * start_x, const Number * start_lam, const Number * start_z_L, const Number * start_z_U, Eval_F_CB eval_f, Eval_G_CB eval_g, Eval_Grad_F_CB eval_grad_f, Eval_Jac_G_CB eval_jac_g, Eval_H_CB eval_h, Intermediate_CB intermediate_cb, Number * x_sol, Number * z_L_sol, Number * z_U_sol, Number * g_sol, Number * lam_sol, Number * obj_sol, UserDataPtr user_data, Number obj_scaling = 1, const Number * x_scaling = NULL, const Number * g_scaling = NULL)`

Constructor, given dimensions of problem, function pointers for evaluation callback functions, and starting points.

Note that the constructor does not make a copy of any of the Number arrays, i.e. it is up to the caller to keep them around.

6.158.2.2 `virtual Ipopt::StdInterfaceTNLP::~~StdInterfaceTNLP () [virtual]`

Default destructor.

6.158.2.3 `Ipopt::StdInterfaceTNLP::StdInterfaceTNLP () [private]`

Default Constructor.

6.158.2.4 `Ipopt::StdInterfaceTNLP::StdInterfaceTNLP (const StdInterfaceTNLP &) [private]`

Copy Constructor.

6.158.3 Member Function Documentation

6.158.3.1 `virtual bool Ipopt::StdInterfaceTNLP::get_nlp_info (Index & n, Index & m, Index & nnz_jac_g, Index & nnz_h_lag, IndexStyleEnum & index_style) [virtual]`

returns dimensions of the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.158.3.2 `virtual bool Ipopt::StdInterfaceTNLP::get_bounds_info (Index n, Number * x_l, Number * x_u, Index m, Number * g_l, Number * g_u) [virtual]`

returns bounds of the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.158.3.3 `virtual bool Ipopt::StdInterfaceTNLP::get_scaling_parameters (Number & obj_scaling, bool & use_x_scaling, Index n, Number * x_scaling, bool & use_g_scaling, Index m, Number * g_scaling) [virtual]`

returns scaling parameters (if `nlp_scaling_method` is selected as user-scaling).

Overloaded from [TNLP](#)

Reimplemented from [Ipopt::TNLP](#).

6.158.3.4 `virtual bool Ipopt::StdInterfaceTNLP::get_starting_point (Index n, bool init_x, Number * x, bool init_z, Number * z_L, Number * z_U, Index m, bool init_lambda, Number * lambda)` [virtual]

provides a starting point for the nlp variables.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.158.3.5 `virtual bool Ipopt::StdInterfaceTNLP::eval_f (Index n, const Number * x, bool new_x, Number & obj_value)` [virtual]

evaluates the objective value for the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.158.3.6 `virtual bool Ipopt::StdInterfaceTNLP::eval_grad_f (Index n, const Number * x, bool new_x, Number * grad_f)` [virtual]

evaluates the gradient of the objective for the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.158.3.7 `virtual bool Ipopt::StdInterfaceTNLP::eval_g (Index n, const Number * x, bool new_x, Index m, Number * g)` [virtual]

evaluates the constraint residuals for the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.158.3.8 `virtual bool Ipopt::StdInterfaceTNLP::eval_jac_g (Index n, const Number * x, bool new_x, Index m, Index nele_jac, Index * iRow, Index * jCol, Number * values)` [virtual]

specifies the jacobian structure (if values is NULL) and evaluates the jacobian values (if values is not NULL) for the nlp.

Overloaded from [TNLP](#)

Implements [Ipopt::TNLP](#).

6.158.3.9 `virtual bool Ipopt::StdInterfaceTNLP::eval_h (Index n, const Number * x, bool new_x, Number obj_factor, Index m, const Number * lambda, bool new_lambda, Index nele_hess, Index * iRow, Index * jCol, Number * values)` [virtual]

specifies the structure of the hessian of the lagrangian (if values is NULL) and evaluates the values (if values is not NULL).

Overloaded from [TNLP](#)

Reimplemented from [Ipopt::TNLP](#).

6.158.3.10 `virtual bool Ipopt::StdInterfaceTNLP::intermediate_callback (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq)` [virtual]

Intermediate Callback method for the user.

Overloaded from [TNLP](#)

Reimplemented from [Ipopt::TNLP](#).

```
6.158.3.11 virtual void Ipopt::StdInterfaceTNLP::finalize_solution ( SolverReturn status, Index n, const Number * x, const
    Number * z_L, const Number * z_U, Index m, const Number * g, const Number * lambda, Number obj_value,
    const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq ) [virtual]
```

This method is called when the algorithm is complete so the [TNLP](#) can store/write the solution.

Implements [Ipopt::TNLP](#).

```
6.158.3.12 void Ipopt::StdInterfaceTNLP::apply_new_x ( bool new_x, Index n, const Number * x ) [private]
```

Internal function to update the internal and ampl state if the x value changes.

```
6.158.3.13 void Ipopt::StdInterfaceTNLP::operator= ( const StdInterfaceTNLP & ) [private]
```

Overloaded Equals Operator.

6.158.4 Member Data Documentation

```
6.158.4.1 SmartPtr<const Journalist> Ipopt::StdInterfaceTNLP::jnlst_ [private]
```

Journalist.

Definition at line 146 of file IpStdInterfaceTNLP.hpp.

```
6.158.4.2 const Index Ipopt::StdInterfaceTNLP::n_var_ [private]
```

Number of variables.

Definition at line 151 of file IpStdInterfaceTNLP.hpp.

```
6.158.4.3 const Index Ipopt::StdInterfaceTNLP::n_con_ [private]
```

Number of constraints.

Definition at line 153 of file IpStdInterfaceTNLP.hpp.

```
6.158.4.4 const Number* Ipopt::StdInterfaceTNLP::x_L_ [private]
```

Pointer to Number array containing lower bounds for variables.

Definition at line 155 of file IpStdInterfaceTNLP.hpp.

```
6.158.4.5 const Number* Ipopt::StdInterfaceTNLP::x_U_ [private]
```

Pointer to Number array containing upper bounds for variables.

Definition at line 157 of file IpStdInterfaceTNLP.hpp.

```
6.158.4.6 const Number* Ipopt::StdInterfaceTNLP::g_L_ [private]
```

Pointer to Number array containing lower bounds for constraints.

Definition at line 159 of file IpStdInterfaceTNLP.hpp.

6.158.4.7 `const Number* Ipopt::StdInterfaceTNLP::g_U_ [private]`

Pointer to Number array containing upper bounds for constraints.

Definition at line 161 of file IpStdInterfaceTNLP.hpp.

6.158.4.8 `const Index Ipopt::StdInterfaceTNLP::nele_jac_ [private]`

Number of non-zero elements in the constraint Jacobian.

Definition at line 163 of file IpStdInterfaceTNLP.hpp.

6.158.4.9 `const Index Ipopt::StdInterfaceTNLP::nele_hess_ [private]`

Number of non-zero elements in the Hessian.

Definition at line 165 of file IpStdInterfaceTNLP.hpp.

6.158.4.10 `const Index Ipopt::StdInterfaceTNLP::index_style_ [private]`

Starting value of the iRow and jCol parameters for matrices.

Definition at line 167 of file IpStdInterfaceTNLP.hpp.

6.158.4.11 `const Number* Ipopt::StdInterfaceTNLP::start_x_ [private]`

Pointer to Number array containing starting point for variables.

Definition at line 169 of file IpStdInterfaceTNLP.hpp.

6.158.4.12 `const Number* Ipopt::StdInterfaceTNLP::start_lam_ [private]`

Pointer to Number array containing starting values for constraint multipliers.

Definition at line 172 of file IpStdInterfaceTNLP.hpp.

6.158.4.13 `const Number* Ipopt::StdInterfaceTNLP::start_z_L_ [private]`

Pointer to Number array containing starting values for lower bound multipliers.

Definition at line 175 of file IpStdInterfaceTNLP.hpp.

6.158.4.14 `const Number* Ipopt::StdInterfaceTNLP::start_z_U_ [private]`

Pointer to Number array containing starting values for upper bound multipliers.

Definition at line 178 of file IpStdInterfaceTNLP.hpp.

6.158.4.15 `Eval_F_CB Ipopt::StdInterfaceTNLP::eval_f_ [private]`

Pointer to callback function evaluating value of objective function.

Definition at line 180 of file IpStdInterfaceTNLP.hpp.

6.158.4.16 `Eval_G_CB Ipopt::StdInterfaceTNLP::eval_g_ [private]`

Pointer to callback function evaluating value of constraints.

Definition at line 182 of file IpStdInterfaceTNLP.hpp.

6.158.4.17 Eval_Grad_F_CB `Ipopt::StdInterfaceTNLP::eval_grad_f_ [private]`

Pointer to callback function evaluating gradient of objective function.

Definition at line 185 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.18 Eval_Jac_G_CB `Ipopt::StdInterfaceTNLP::eval_jac_g_ [private]`

Pointer to callback function evaluating Jacobian of constraints.

Definition at line 187 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.19 Eval_H_CB `Ipopt::StdInterfaceTNLP::eval_h_ [private]`

Pointer to callback function evaluating Hessian of Lagrangian.

Definition at line 189 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.20 Intermediate_CB `Ipopt::StdInterfaceTNLP::intermediate_cb_ [private]`

Pointer to intermediate callback function giving control to user.

Definition at line 191 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.21 UserDataPtr `Ipopt::StdInterfaceTNLP::user_data_ [private]`

Pointer to user data.

Definition at line 193 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.22 Number `Ipopt::StdInterfaceTNLP::obj_scaling_ [private]`

Objective scaling factor.

Definition at line 195 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.23 const Number* `Ipopt::StdInterfaceTNLP::x_scaling_ [private]`

Scaling factors for variables (if not NULL)

Definition at line 197 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.24 const Number* `Ipopt::StdInterfaceTNLP::g_scaling_ [private]`

Scaling factors for constraints (if not NULL)

Definition at line 199 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.25 Number* `Ipopt::StdInterfaceTNLP::non_const_x_ [private]`

A non-const copy of `x` - this is kept up-to-date in `apply_new_x`.

Definition at line 204 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.26 Number* `Ipopt::StdInterfaceTNLP::x_sol_ [private]`

Pointers to the user provided vectors for solution.

Definition at line 207 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.27 **Number*** `Ipopt::StdInterfaceTNLP::z_L_sol_` `[private]`

Definition at line 208 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.28 **Number*** `Ipopt::StdInterfaceTNLP::z_U_sol_` `[private]`

Definition at line 209 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.29 **Number*** `Ipopt::StdInterfaceTNLP::g_sol_` `[private]`

Definition at line 210 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.30 **Number*** `Ipopt::StdInterfaceTNLP::lambda_sol_` `[private]`

Definition at line 211 of file `IpStdInterfaceTNLP.hpp`.

6.158.4.31 **Number*** `Ipopt::StdInterfaceTNLP::obj_sol_` `[private]`

Definition at line 212 of file `IpStdInterfaceTNLP.hpp`.

The documentation for this class was generated from the following file:

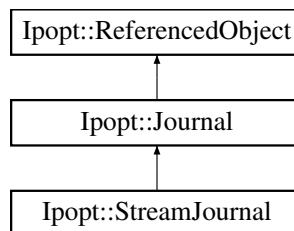
- [Interfaces/IpStdInterfaceTNLP.hpp](#)

6.159 Ipopt::StreamJournal Class Reference

[StreamJournal](#) class.

```
#include <IpJournalist.hpp>
```

Inheritance diagram for `Ipopt::StreamJournal`:



Public Member Functions

- [StreamJournal](#) (const std::string &name, [EJournalLevel](#) default_level)
Constructor.
- virtual [~StreamJournal](#) ()
Destructor.
- void [SetOutputStream](#) (std::ostream *os)
Setting the output stream pointer.

Protected Member Functions

Implementation version of Print methods - Overloaded from
[Journal](#) base class.

- virtual void `PrintImpl` (`EJournalCategory` category, `EJournalLevel` level, const char *str)
Print to the designated output location.
- virtual void `PrintfImpl` (`EJournalCategory` category, `EJournalLevel` level, const char *pformat, va_list ap)
Printf to the designated output location.
- virtual void `FlushBufferImpl` ()
Flush output buffer.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `StreamJournal` ()
Default Constructor.
- `StreamJournal` (const `StreamJournal` &)
Copy Constructor.
- void `operator=` (const `StreamJournal` &)
Overloaded Equals Operator.

Private Attributes

- std::ostream * `os_`
pointer to output stream for the output destination
- char `buffer_` [32768]
buffer for sprintf.

6.159.1 Detailed Description

`StreamJournal` class.

This is a particular `Journal` implementation that writes to a stream for output.

Definition at line 440 of file `IpJournalist.hpp`.

6.159.2 Constructor & Destructor Documentation

6.159.2.1 `Ipopt::StreamJournal::StreamJournal (const std::string & name, EJournalLevel default_level)`

Constructor.

6.159.2.2 `virtual Ipopt::StreamJournal::~StreamJournal () [inline],[virtual]`

Destructor.

Definition at line 447 of file `IpJournalist.hpp`.

6.159.2.3 `Ipopt::StreamJournal::StreamJournal () [private]`

Default Constructor.

6.159.2.4 Ipopt::StreamJournal::StreamJournal (const StreamJournal &) [private]

Copy Constructor.

6.159.3 Member Function Documentation

6.159.3.1 void Ipopt::StreamJournal::SetOutputStream (std::ostream * os)

Setting the output stream pointer.

6.159.3.2 virtual void Ipopt::StreamJournal::PrintImpl (EJournalCategory category, EJournalLevel level, const char * str)
[protected], [virtual]

Print to the designated output location.

Implements [Ipopt::Journal](#).

6.159.3.3 virtual void Ipopt::StreamJournal::PrintfImpl (EJournalCategory category, EJournalLevel level, const char *
pformat, va_list ap) [protected], [virtual]

Printf to the designated output location.

Implements [Ipopt::Journal](#).

6.159.3.4 virtual void Ipopt::StreamJournal::FlushBufferImpl () [protected], [virtual]

Flush output buffer.

Implements [Ipopt::Journal](#).

6.159.3.5 void Ipopt::StreamJournal::operator= (const StreamJournal &) [private]

Overloaded Equals Operator.

6.159.4 Member Data Documentation

6.159.4.1 std::ostream* Ipopt::StreamJournal::os_ [private]

pointer to output stream for the output destination

Definition at line 490 of file IpJournalist.hpp.

6.159.4.2 char Ipopt::StreamJournal::buffer_[32768] [private]

buffer for sprintf.

Being generous in size here...

Definition at line 493 of file IpJournalist.hpp.

The documentation for this class was generated from the following file:

- Common/[IpJournalist.hpp](#)

6.160 Ipopt::RegisteredOption::string_entry Class Reference

class to hold the valid string settings for a string option

```
#include <IpRegOptions.hpp>
```

Public Member Functions

- [string_entry](#) (const std::string &value, const std::string &description)

Public Attributes

- std::string [value_](#)
- std::string [description_](#)

6.160.1 Detailed Description

class to hold the valid string settings for a string option

Definition at line 37 of file IpRegOptions.hpp.

6.160.2 Constructor & Destructor Documentation

6.160.2.1 `Ipopt::RegisteredOption::string_entry::string_entry (const std::string & value, const std::string & description)`
`[inline]`

Definition at line 40 of file IpRegOptions.hpp.

6.160.3 Member Data Documentation

6.160.3.1 `std::string Ipopt::RegisteredOption::string_entry::value_`

Definition at line 43 of file IpRegOptions.hpp.

6.160.3.2 `std::string Ipopt::RegisteredOption::string_entry::description_`

Definition at line 44 of file IpRegOptions.hpp.

The documentation for this class was generated from the following file:

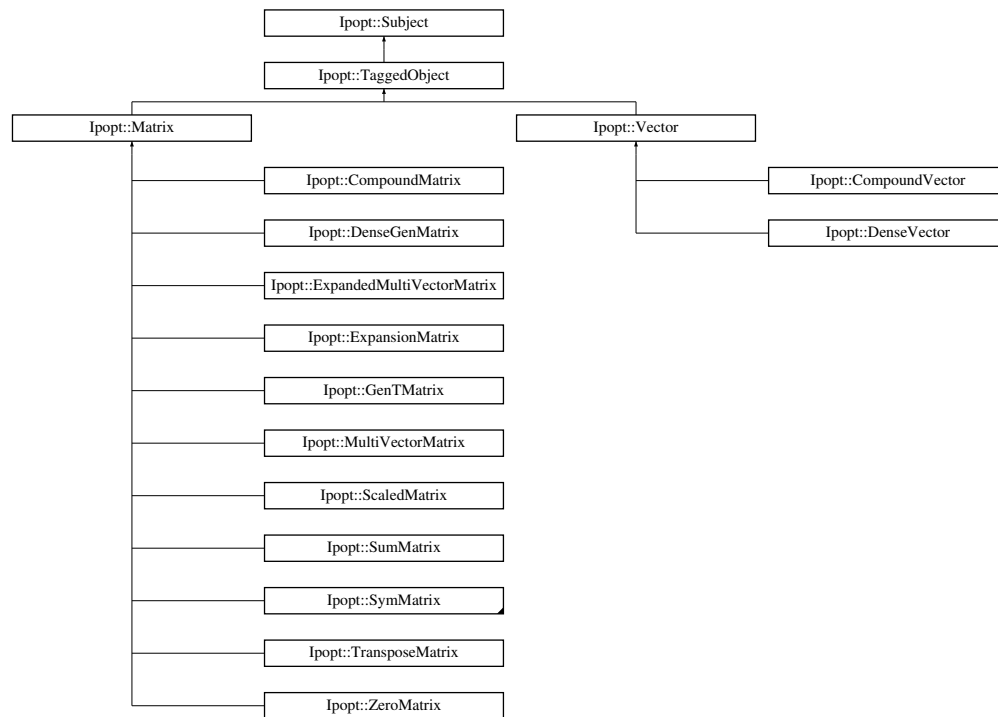
- [Common/IpRegOptions.hpp](#)

6.161 Ipopt::Subject Class Reference

Slight Variation of the [Observer](#) Design Pattern ([Subject](#) part).

```
#include <IpObserver.hpp>
```

Inheritance diagram for Ipopt::Subject:



Public Member Functions

Constructors/Destructors

- [Subject](#) ()
Default Constructor.
- virtual [~Subject](#) ()
Default destructor.

Methods to Add and Remove Observers.

Currently, the `notify_type` flags are not used, and Observers are attached in general and will receive all notifications (of the type requested and possibly of types not requested).

It is up to the observer to ignore the types they are not interested in. The `NotifyType` in the parameter list is so a more efficient mechanism depending on type could be implemented later if necessary.

- void [AttachObserver](#) ([Observer::NotifyType](#) notify_type, [Observer](#) *observer) const
Attach the specified observer (i.e., begin receiving notifications).
- void [DetachObserver](#) ([Observer::NotifyType](#) notify_type, [Observer](#) *observer) const
Detach the specified observer (i.e., no longer receive notifications).

Protected Member Functions

- void [Notify](#) ([Observer::NotifyType](#) notify_type) const

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `Subject` (const `Subject` &)
Copy Constructor.
- `void operator=` (const `Subject` &)
Overloaded Equals Operator.

Private Attributes

- `std::vector< Observer * > observers_`

6.161.1 Detailed Description

Slight Variation of the `Observer` Design Pattern (`Subject` part).

This class implements the `Subject` class of the `Observer` Design Pattern. An `Observer` "Attach"es to a `Subject`, indicating that it would like to be notified of changes in the `Subject`. Any derived class that is to be observed has to inherit off the `Subject` base class. If the subject needs to notify the `Observer`, it calls the `Notify` method.

Definition at line 129 of file `IpObserver.hpp`.

6.161.2 Constructor & Destructor Documentation

6.161.2.1 `Ipopt::Subject::Subject () [inline]`

Default Constructor.

Definition at line 140 of file `IpObserver.hpp`.

6.161.2.2 `Ipopt::Subject::~~Subject () [inline], [virtual]`

Default destructor.

Definition at line 296 of file `IpObserver.hpp`.

6.161.2.3 `Ipopt::Subject::Subject (const Subject &) [private]`

Copy Constructor.

6.161.3 Member Function Documentation

6.161.3.1 `void Ipopt::Subject::AttachObserver (Observer::NotifyType notify_type, Observer * observer) const [inline]`

Attach the specified observer (i.e., begin receiving notifications).

Definition at line 309 of file `IpObserver.hpp`.

6.161.3.2 `void Ipopt::Subject::DetachObserver (Observer::NotifyType notify_type, Observer * observer) const [inline]`

Detach the specified observer (i.e., no longer receive notifications).

Definition at line 329 of file `IpObserver.hpp`.

6.161.3.3 void Ipopt::Subject::Notify (Observer::NotifyType *notify_type*) const [inline], [protected]

Definition at line 351 of file IpObserver.hpp.

6.161.3.4 void Ipopt::Subject::operator= (const Subject &) [private]

Overloaded Equals Operator.

6.161.4 Member Data Documentation

6.161.4.1 std::vector<Observer*> Ipopt::Subject::observers_ [mutable], [private]

Definition at line 192 of file IpObserver.hpp.

The documentation for this class was generated from the following file:

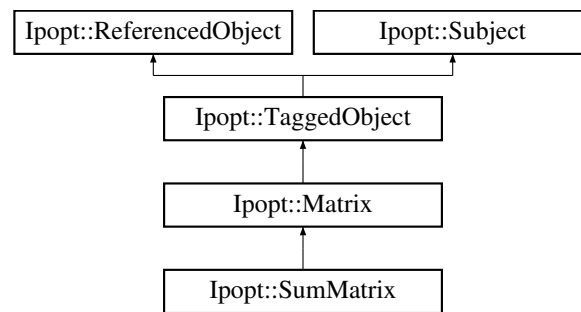
- [Common/IpObserver.hpp](#)

6.162 Ipopt::SumMatrix Class Reference

Class for Matrices which are sum of matrices.

```
#include <IpSumMatrix.hpp>
```

Inheritance diagram for Ipopt::SumMatrix:



Public Member Functions

- void [SetTerm](#) (Index item, [Number](#) factor, const [Matrix](#) &matrix)
Method for setting term item for the sum.
- void [GetTerm](#) (Index item, [Number](#) &factor, [SmartPtr](#)< const [Matrix](#) > &matrix) const
Method for getting term item for the sum.
- [Index](#) [NTerms](#) () const
Return the number of terms.

Constructors / Destructors

- [SumMatrix](#) (const [SumMatrixSpace](#) *owner_space)
Constructor, taking the owner_space.
- virtual [~SumMatrix](#) ()
Destructor.

Protected Member Functions

Methods overloaded from matrix

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual void [TransMultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix(transpose) vector multiply.
- virtual bool [IsValidNumbersImpl](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &cols_norms, bool init) const
Compute the max-norm of the columns in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [SumMatrix](#) ()
Default Constructor.
- [SumMatrix](#) (const [SumMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [SumMatrix](#) &)
Overloaded Equals Operator.

Private Attributes

- std::vector< [Number](#) > [factors_](#)
std::vector storing the factors for each term.
- std::vector< [SmartPtr](#)< const [Matrix](#) > > [matrices_](#)
std::vector storing the matrices for each term.
- const [SumMatrixSpace](#) * [owner_space_](#)
Copy of the owner_space as a [SumMatrixSpace](#).

Additional Inherited Members

6.162.1 Detailed Description

Class for Matrices which are sum of matrices.

For each term in the we store the matrix and a factor.

Definition at line 24 of file [IpSumMatrix.hpp](#).

6.162.2 Constructor & Destructor Documentation

6.162.2.1 Ipopt::SumMatrix::SumMatrix (const SumMatrixSpace * owner_space)

Constructor, taking the owner_space.

6.162.2.2 virtual Ipopt::SumMatrix::~~SumMatrix () [virtual]

Destructor.

6.162.2.3 Ipopt::SumMatrix::SumMatrix () [private]

Default Constructor.

6.162.2.4 Ipopt::SumMatrix::SumMatrix (const SumMatrix &) [private]

Copy Constructor.

6.162.3 Member Function Documentation

6.162.3.1 void Ipopt::SumMatrix::SetTerm (Index item, Number factor, const Matrix & matrix)

Method for setting term item for the sum.

6.162.3.2 void Ipopt::SumMatrix::GetTerm (Index item, Number & factor, SmartPtr< const Matrix > & matrix) const

Method for getting term item for the sum.

Note that counting of terms starts at 0.

6.162.3.3 Index Ipopt::SumMatrix::NTerms () const

Return the number of terms.

6.162.3.4 virtual void Ipopt::SumMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.162.3.5 virtual void Ipopt::SumMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.162.3.6 virtual bool Ipopt::SumMatrix::IsValidNumbersImpl () const [protected], [virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.162.3.7 `virtual void Ipopt::SumMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` [protected],
[virtual]

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.162.3.8 `virtual void Ipopt::SumMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const` [protected],
[virtual]

Compute the max-norm of the columns in the matrix.

The result is stored in cols_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.162.3.9 `virtual void Ipopt::SumMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected], [virtual]

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.162.3.10 `void Ipopt::SumMatrix::operator= (const SumMatrix &)` [private]

Overloaded Equals Operator.

6.162.4 Member Data Documentation

6.162.4.1 `std::vector<Number> Ipopt::SumMatrix::factors_` [private]

std::vector storing the factors for each term.

Definition at line 93 of file IpSumMatrix.hpp.

6.162.4.2 `std::vector<SmartPointer<const Matrix>> Ipopt::SumMatrix::matrices_` [private]

std::vector storing the matrices for each term.

Definition at line 96 of file IpSumMatrix.hpp.

6.162.4.3 `const SumMatrixSpace* Ipopt::SumMatrix::owner_space_` [private]

Copy of the owner_space as a [SumMatrixSpace](#).

Definition at line 99 of file IpSumMatrix.hpp.

The documentation for this class was generated from the following file:

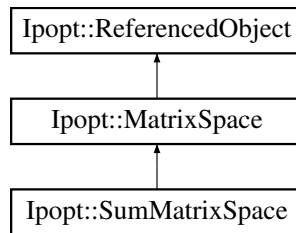
- [LinAlg/IpSumMatrix.hpp](#)

6.163 Ipopt::SumMatrixSpace Class Reference

Class for matrix space for [SumMatrix](#).

```
#include <IpSumMatrix.hpp>
```

Inheritance diagram for Ipopt::SumMatrixSpace:



Public Member Functions

- [Index](#) [NTerms](#) () const
Accessor functions to get the number of terms in the sum.
- void [SetTermSpace](#) ([Index](#) term_idx, const [MatrixSpace](#) &mat_space)
Set the appropriate matrix space for each term.
- [SmartPtr](#)< const [MatrixSpace](#) > [GetTermSpace](#) ([Index](#) term_idx) const
Get the matrix space for a particular term.
- [SumMatrix](#) * [MakeNewSumMatrix](#) () const
Method for creating a new matrix of this specific type.
- virtual [Matrix](#) * [MakeNew](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.

Constructors / Destructors

- [SumMatrixSpace](#) ([Index](#) nrow, [Index](#) ncol, [Index](#) nterms)
Constructor, given the number of row and columns, as well as the number of terms in the sum.
- virtual [~SumMatrixSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [SumMatrixSpace](#) ()
Default constructor.
- [SumMatrixSpace](#) (const [SumMatrixSpace](#) &)
Copy Constructor.
- [SumMatrixSpace](#) & [operator=](#) (const [SumMatrixSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- const [Index](#) nterms_
- std::vector< [SmartPtr](#)< const [MatrixSpace](#) > > term_spaces_

6.163.1 Detailed Description

Class for matrix space for [SumMatrix](#).

Definition at line 103 of file `IpSumMatrix.hpp`.

6.163.2 Constructor & Destructor Documentation

6.163.2.1 `Ipopt::SumMatrixSpace::SumMatrixSpace (Index nrows, Index ncols, Index nterms) [inline]`

Constructor, given the number of row and columns, as well as the number of terms in the sum.

Definition at line 111 of file `IpSumMatrix.hpp`.

6.163.2.2 `virtual Ipopt::SumMatrixSpace::~~SumMatrixSpace () [inline],[virtual]`

Destructor.

Definition at line 118 of file `IpSumMatrix.hpp`.

6.163.2.3 `Ipopt::SumMatrixSpace::SumMatrixSpace () [private]`

Default constructor.

6.163.2.4 `Ipopt::SumMatrixSpace::SumMatrixSpace (const SumMatrixSpace &) [private]`

Copy Constructor.

6.163.3 Member Function Documentation

6.163.3.1 `Index Ipopt::SumMatrixSpace::NTerms () const [inline]`

Accessor functions to get the number of terms in the sum.

Definition at line 123 of file `IpSumMatrix.hpp`.

6.163.3.2 `void Ipopt::SumMatrixSpace::SetTermSpace (Index term_idx, const MatrixSpace & mat_space)`

Set the appropriate matrix space for each term.

This must be called for each term or a runtime error will occur

6.163.3.3 `SmartPtr<const MatrixSpace> Ipopt::SumMatrixSpace::GetTermSpace (Index term_idx) const`

Get the matrix space for a particular term.

6.163.3.4 `SumMatrix* Ipopt::SumMatrixSpace::MakeNewSumMatrix () const`

Method for creating a new matrix of this specific type.

6.163.3.5 `virtual Matrix* Ipopt::SumMatrixSpace::MakeNew () const [virtual]`

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

6.163.3.6 SumMatrixSpace& Ipopt::SumMatrixSpace::operator= (const SumMatrixSpace &) [private]

Overloaded Equals Operator.

6.163.4 Member Data Documentation

6.163.4.1 const Index Ipopt::SumMatrixSpace::nterms_ [private]

Definition at line 161 of file IpSumMatrix.hpp.

6.163.4.2 std::vector< SmartPtr<const MatrixSpace> > Ipopt::SumMatrixSpace::term_spaces_ [private]

Definition at line 163 of file IpSumMatrix.hpp.

The documentation for this class was generated from the following file:

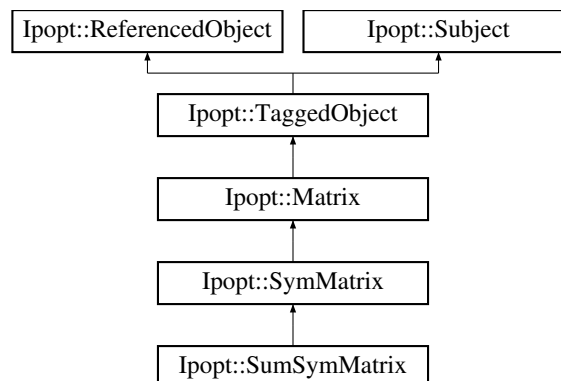
- [LinAlg/IpSumMatrix.hpp](#)

6.164 Ipopt::SumSymMatrix Class Reference

Class for Matrices which are sum of symmetric matrices.

```
#include <IpSumSymMatrix.hpp>
```

Inheritance diagram for Ipopt::SumSymMatrix:



Public Member Functions

- void [SetTerm](#) (Index item, Number factor, const [SymMatrix](#) &matrix)
Method for setting term item for the sum.
- void [GetTerm](#) (Index item, Number &factor, SmartPtr< const [SymMatrix](#) > &matrix) const
Method for getting term item for the sum.
- Index [NTerms](#) () const
Return the number of terms.

Constructors / Destructors

- [SumSymMatrix](#) (const [SumSymMatrixSpace](#) *owner_space)
Constructor, initializing with dimensions of the matrix and the number of terms in the sum.
- [~SumSymMatrix](#) ()
Destructor.

Protected Member Functions

Methods overloaded from matrix

- virtual void `MultVectorImpl` (`Number` alpha, const `Vector` &x, `Number` beta, `Vector` &y) const
Matrix-vector multiply.
- virtual bool `IsValidNumbersImpl` () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void `ComputeRowAMaxImpl` (`Vector` &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void `ComputeColAMaxImpl` (`Vector` &cols_norms, bool init) const
Since the matrix is symmetric, the row and column max norms are identical.
- virtual void `PrintImpl` (const `Journalist` &jnlst, `EJournalLevel` level, `EJournalCategory` category, const std::string &name, `Index` indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `SumSymMatrix` ()
Default Constructor.
- `SumSymMatrix` (const `SumSymMatrix` &)
Copy Constructor.
- void `operator=` (const `SumSymMatrix` &)
Overloaded Equals Operator.

Private Attributes

- std::vector< `Number` > `factors_`
std::vector storing the factors for each term.
- std::vector< `SmartPtr`< const `SymMatrix` > > `matrices_`
std::vector storing the matrices for each term.
- const `SumSymMatrixSpace` * `owner_space_`
Copy of the owner_space as a SumSymMatrixSpace.

Additional Inherited Members

6.164.1 Detailed Description

Class for Matrices which are sum of symmetric matrices.

For each term in the we store the matrix and a factor.

Definition at line 24 of file `IpSumSymMatrix.hpp`.

6.164.2 Constructor & Destructor Documentation

6.164.2.1 Ipopt::SumSymMatrix::SumSymMatrix (const SumSymMatrixSpace * owner_space)

Constructor, initializing with dimensions of the matrix and the number of terms in the sum.

6.164.2.2 Ipopt::SumSymMatrix::~~SumSymMatrix ()

Destructor.

6.164.2.3 Ipopt::SumSymMatrix::SumSymMatrix () [private]

Default Constructor.

6.164.2.4 Ipopt::SumSymMatrix::SumSymMatrix (const SumSymMatrix &) [private]

Copy Constructor.

6.164.3 Member Function Documentation

6.164.3.1 void Ipopt::SumSymMatrix::SetTerm (Index item, Number factor, const SymMatrix & matrix)

Method for setting term item for the sum.

Note that counting of terms starts at 0.

6.164.3.2 void Ipopt::SumSymMatrix::GetTerm (Index item, Number & factor, SmartPtr< const SymMatrix > & matrix) const

Method for getting term item for the sum.

Note that counting of terms starts at 0.

6.164.3.3 Index Ipopt::SumSymMatrix::NTerms () const

Return the number of terms.

6.164.3.4 virtual void Ipopt::SumSymMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected],[virtual]

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.164.3.5 virtual bool Ipopt::SumSymMatrix::IsValidNumbersImpl () const [protected],[virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.164.3.6 virtual void Ipopt::SumSymMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const [protected],[virtual]

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.164.3.7 `virtual void Ipopt::SumSymMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const` [protected], [virtual]

Since the matrix is symmetric, the row and column max norms are identical.

Reimplemented from [Ipopt::SymMatrix](#).

6.164.3.8 `virtual void Ipopt::SumSymMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected], [virtual]

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.164.3.9 `void Ipopt::SumSymMatrix::operator= (const SumSymMatrix &)` [private]

Overloaded Equals Operator.

6.164.4 Member Data Documentation

6.164.4.1 `std::vector<Number> Ipopt::SumSymMatrix::factors_` [private]

std::vector storing the factors for each term.

Definition at line 93 of file IpSumSymMatrix.hpp.

6.164.4.2 `std::vector<SmartPointer<const SymMatrix> > Ipopt::SumSymMatrix::matrices_` [private]

std::vector storing the matrices for each term.

Definition at line 96 of file IpSumSymMatrix.hpp.

6.164.4.3 `const SumSymMatrixSpace* Ipopt::SumSymMatrix::owner_space_` [private]

Copy of the owner_space as a [SumSymMatrixSpace](#).

Definition at line 99 of file IpSumSymMatrix.hpp.

The documentation for this class was generated from the following file:

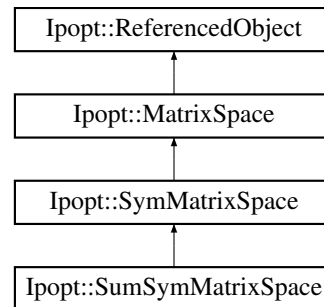
- [LinAlg/IpSumSymMatrix.hpp](#)

6.165 Ipopt::SumSymMatrixSpace Class Reference

Class for matrix space for [SumSymMatrix](#).

```
#include <IpSumSymMatrix.hpp>
```

Inheritance diagram for Ipopt::SumSymMatrixSpace:



Public Member Functions

- void [SetTermSpace](#) ([Index](#) term_idx, const [SymMatrixSpace](#) &space)
Use this method to set the matrix spaces for the various terms.
- [SmartPtr](#)< const [SymMatrixSpace](#) > [GetTermSpace](#) ([Index](#) term_idx) const
Get the matrix space for a particular term.
- [SumSymMatrix](#) * [MakeNewSumSymMatrix](#) () const
Method for creating a new matrix of this specific type.
- virtual [SymMatrix](#) * [MakeNewSymMatrix](#) () const
Overloaded MakeNew method for the [SymMatrixSpace](#) base class.

Constructors / Destructors

- [SumSymMatrixSpace](#) ([Index](#) ndim, [Index](#) nterms)
Constructor, given the dimension of the matrix and the number of terms in the sum.
- [~SumSymMatrixSpace](#) ()
Destructor.

Accessor functions

- [Index](#) [NTerms](#) () const
Number of terms in the sum.

Private Attributes

- [Index](#) [nterms_](#)
- std::vector< [SmartPtr](#)< const [SymMatrixSpace](#) > > [term_spaces_](#)

6.165.1 Detailed Description

Class for matrix space for [SumSymMatrix](#).

Definition at line 103 of file [IpSumSymMatrix.hpp](#).

6.165.2 Constructor & Destructor Documentation

6.165.2.1 Ipopt::SumSymMatrixSpace::SumSymMatrixSpace ([Index](#) ndim, [Index](#) nterms) [\[inline\]](#)

Constructor, given the dimension of the matrix and the number of terms in the sum.

Definition at line 110 of file [IpSumSymMatrix.hpp](#).

6.165.2.2 `Ipopt::SumSymMatrixSpace::~~SumSymMatrixSpace () [inline]`

Destructor.

Definition at line 117 of file `IpSumSymMatrix.hpp`.

6.165.3 Member Function Documentation

6.165.3.1 `Index Ipopt::SumSymMatrixSpace::NTerms () const [inline]`

Number of terms in the sum.

Definition at line 124 of file `IpSumSymMatrix.hpp`.

6.165.3.2 `void Ipopt::SumSymMatrixSpace::SetTermSpace (Index term_idx, const SymMatrixSpace & space)`

Use this method to set the matrix spaces for the various terms.

You will not be able to create a matrix until all these spaces are set.

6.165.3.3 `SmartPtr<const SymMatrixSpace> Ipopt::SumSymMatrixSpace::GetTermSpace (Index term_idx) const`

Get the matrix space for a particular term.

6.165.3.4 `SumSymMatrix* Ipopt::SumSymMatrixSpace::MakeNewSumSymMatrix () const`

Method for creating a new matrix of this specific type.

6.165.3.5 `virtual SymMatrix* Ipopt::SumSymMatrixSpace::MakeNewSymMatrix () const [virtual]`

Overloaded MakeNew method for the [SymMatrixSpace](#) base class.

Implements [Ipopt::SymMatrixSpace](#).

6.165.4 Member Data Documentation

6.165.4.1 `Index Ipopt::SumSymMatrixSpace::nterms_ [private]`

Definition at line 146 of file `IpSumSymMatrix.hpp`.

6.165.4.2 `std::vector< SmartPtr<const SymMatrixSpace> > Ipopt::SumSymMatrixSpace::term_spaces_ [private]`

Definition at line 148 of file `IpSumSymMatrix.hpp`.

The documentation for this class was generated from the following file:

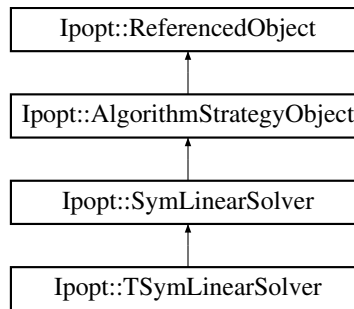
- [LinAlg/IpSumSymMatrix.hpp](#)

6.166 Ipopt::SymLinearSolver Class Reference

Base class for all derived symmetric linear solvers.

`#include <IpSymLinearSolver.hpp>`

Inheritance diagram for `Ipopt::SymLinearSolver`:



Public Member Functions

- virtual bool `InitializeImpl` (const `OptionsList` &options, const std::string &prefix)=0
overloaded from `AlgorithmStrategyObject`

Constructor/Destructor

- `SymLinearSolver` ()
- virtual `~SymLinearSolver` ()

Methods for requesting solution of the linear system.

- virtual `ESymSolverStatus MultiSolve` (const `SymMatrix` &A, std::vector< `SmartPtr`< const `Vector` > > &rhsV, std::vector< `SmartPtr`< `Vector` > > &solV, bool check_NegEVals, `Index` numberOfNegEVals)=0
Solve operation for multiple right hand sides.
- `ESymSolverStatus Solve` (const `SymMatrix` &A, const `Vector` &rhs, `Vector` &sol, bool check_NegEVals, `Index` numberOfNegEVals)
Solve operation for a single right hand side.
- virtual `Index NumberOfNegEVals` () const =0
Number of negative eigenvalues detected during last factorization.
- virtual bool `IncreaseQuality` ()=0
Request to increase quality of solution for next solve.
- virtual bool `ProvidesInertia` () const =0
Query whether inertia is computed by linear solver.

Additional Inherited Members

6.166.1 Detailed Description

Base class for all derived symmetric linear solvers.

In the full space version of `Ipopt` a large linear system has to be solved for the augmented system. This case is meant to be the base class for all derived linear solvers for symmetric matrices (of type `SymMatrix`).

A linear solver can be used repeatedly for matrices with identical structure of nonzero elements. The nonzero structure of those matrices must not be changed between calls.

The called might ask the solver to only solve the linear system if the system is nonsingular, and if the number of negative eigenvalues matches a given number.

Definition at line 50 of file `IpSymLinearSolver.hpp`.

6.166.2 Constructor & Destructor Documentation

6.166.2.1 `Ipopt::SymLinearSolver::SymLinearSolver () [inline]`

Definition at line 55 of file `IpSymLinearSolver.hpp`.

6.166.2.2 `virtual Ipopt::SymLinearSolver::~~SymLinearSolver () [inline],[virtual]`

Definition at line 58 of file `IpSymLinearSolver.hpp`.

6.166.3 Member Function Documentation

6.166.3.1 `virtual bool Ipopt::SymLinearSolver::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::TSymLinearSolver](#).

6.166.3.2 `virtual ESymSolverStatus Ipopt::SymLinearSolver::MultiSolve (const SymMatrix & A, std::vector< SmartPtr< Vector > > & rhsV, std::vector< SmartPtr< Vector > > & solV, bool check_NegEvals, Index numberOfNegEvals) [pure virtual]`

Solve operation for multiple right hand sides.

Solves the linear system $A * Sol = Rhs$ with multiple right hand sides. If necessary, A is factorized. Correct solutions are only guaranteed if the return values is `SYMSOLVER_SUCCESS`. The solver will return `SYMSOLVER_SINGULAR` if the linear system is singular, and it will return `SYMSOLVER_WRONG_INERTIA` if `check_NegEvals` is true and the number of negative eigenvalues in the matrix does not match `numberOfNegEvals`.

`check_NegEvals` cannot be chosen true, if [ProvidesInertia\(\)](#) returns false.

Implemented in [Ipopt::TSymLinearSolver](#).

6.166.3.3 `ESymSolverStatus Ipopt::SymLinearSolver::Solve (const SymMatrix & A, const Vector & rhs, Vector & sol, bool check_NegEvals, Index numberOfNegEvals) [inline]`

Solve operation for a single right hand side.

Solves the linear system $A * Sol = Rhs$. See `MultiSolve` for more details.

Definition at line 89 of file `IpSymLinearSolver.hpp`.

6.166.3.4 `virtual Index Ipopt::SymLinearSolver::NumberOfNegEvals () const [pure virtual]`

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix. This must not be called if the linear solver does not compute this quantities (see [ProvidesInertia](#)).

Implemented in [Ipopt::TSymLinearSolver](#).

6.166.3.5 `virtual bool Ipopt::SymLinearSolver::IncreaseQuality () [pure virtual]`

Request to increase quality of solution for next solve.

Ask linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implemented in [Ipopt::TSymLinearSolver](#).

6.166.3.6 `virtual bool Ipopt::SymLinearSolver::ProvidesInertia () const [pure virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implemented in [Ipopt::TSymLinearSolver](#).

The documentation for this class was generated from the following file:

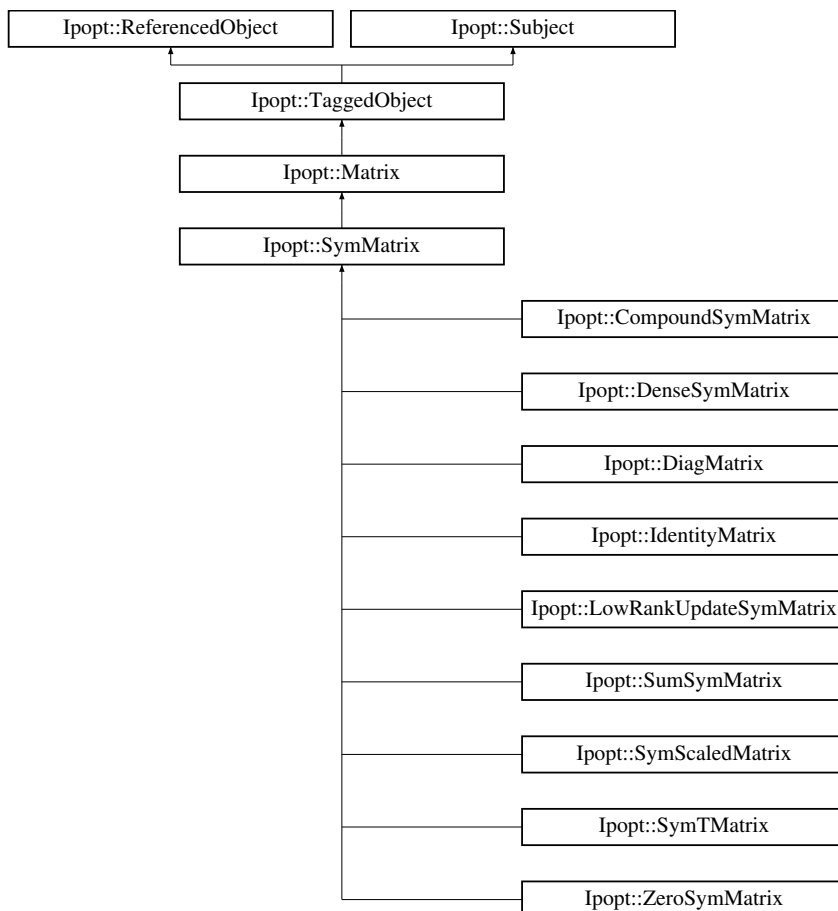
- [Algorithm/LinearSolvers/IpSymLinearSolver.hpp](#)

6.167 Ipopt::SymMatrix Class Reference

This is the base class for all derived symmetric matrix types.

```
#include <IpSymMatrix.hpp>
```

Inheritance diagram for Ipopt::SymMatrix:



Public Member Functions

- [SmartPtr< const SymMatrixSpace > OwnerSymMatrixSpace \(\) const](#)

Constructor/Destructor

- [SymMatrix](#) (const [SymMatrixSpace](#) *owner_space)
Constructor, taking the owner_space.
- virtual [~SymMatrix](#) ()
Destructor.

Information about the size of the matrix

- [Index Dim](#) () const
Dimension of the matrix (number of rows and columns)

Protected Member Functions

Overloaded methods from Matrix.

- virtual void [TransMultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Since the matrix is symmetric, it is only necessary to implement the MultVectorImpl method in a class that inherits from this base class.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &cols_norms, bool init) const
Since the matrix is symmetric, the row and column max norms are identical.

Private Attributes

- const [SymMatrixSpace](#) * owner_space_
Copy of the owner space ptr as a [SymMatrixSpace](#) instead of a [MatrixSpace](#).

Additional Inherited Members

6.167.1 Detailed Description

This is the base class for all derived symmetric matrix types.

Definition at line 23 of file IpSymMatrix.hpp.

6.167.2 Constructor & Destructor Documentation

6.167.2.1 Ipopt::SymMatrix::SymMatrix (const SymMatrixSpace * owner_space) [inline]

Constructor, taking the owner_space.

Definition at line 142 of file IpSymMatrix.hpp.

6.167.2.2 virtual Ipopt::SymMatrix::~SymMatrix () [inline],[virtual]

Destructor.

Definition at line 34 of file IpSymMatrix.hpp.

6.167.3 Member Function Documentation

6.167.3.1 Index Ipopt::SymMatrix::Dim () const [inline]

Dimension of the matrix (number of rows and columns)

Definition at line 149 of file IpSymMatrix.hpp.

6.167.3.2 SmartPtr< const SymMatrixSpace > Ipopt::SymMatrix::OwnerSymMatrixSpace () const [inline]

Definition at line 155 of file IpSymMatrix.hpp.

6.167.3.3 virtual void Ipopt::SymMatrix::TransMultVectorImpl (Number *alpha*, const Vector & *x*, Number *beta*, Vector & *y*) const [inline], [protected], [virtual]

Since the matrix is symmetric, it is only necessary to implement the MultVectorImpl method in a class that inherits from this base class.

If the TransMultVectorImpl is called, this base class automatically calls MultVectorImpl instead.

Implements [Ipopt::Matrix](#).

Reimplemented in [Ipopt::ZeroSymMatrix](#).

Definition at line 56 of file IpSymMatrix.hpp.

6.167.3.4 virtual void Ipopt::SymMatrix::ComputeColAMaxImpl (Vector & *cols_norms*, bool *init*) const [inline], [protected], [virtual]

Since the matrix is symmetric, the row and column max norms are identical.

Implements [Ipopt::Matrix](#).

Reimplemented in [Ipopt::LowRankUpdateSymMatrix](#), [Ipopt::SumSymMatrix](#), and [Ipopt::ZeroSymMatrix](#).

Definition at line 65 of file IpSymMatrix.hpp.

6.167.4 Member Data Documentation

6.167.4.1 const SymMatrixSpace* Ipopt::SymMatrix::owner_space_ [private]

Copy of the owner space ptr as a [SymMatrixSpace](#) instead of a [MatrixSpace](#).

Definition at line 75 of file IpSymMatrix.hpp.

The documentation for this class was generated from the following file:

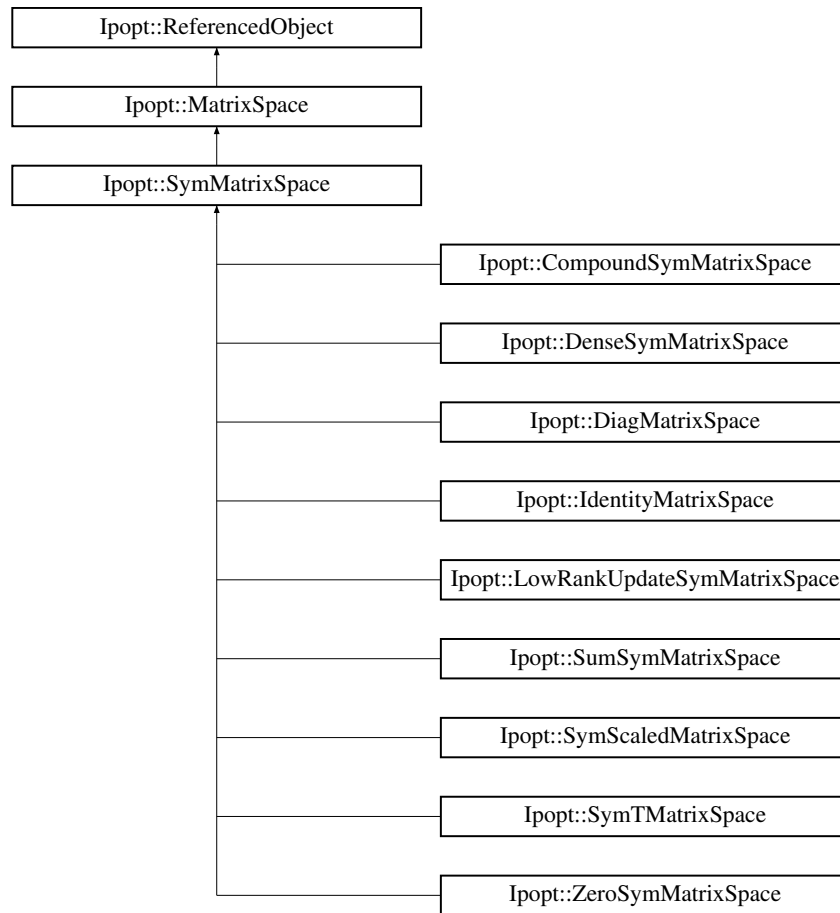
- [LinAlg/IpSymMatrix.hpp](#)

6.168 Ipopt::SymMatrixSpace Class Reference

[SymMatrixSpace](#) base class, corresponding to the [SymMatrix](#) base class.

```
#include <IpSymMatrix.hpp>
```

Inheritance diagram for Ipopt::SymMatrixSpace:



Public Member Functions

- virtual `SymMatrix * MakeNewSymMatrix () const =0`
Pure virtual method for creating a new matrix of this specific type.
- virtual `Matrix * MakeNew () const`
Overloaded MakeNew method for the `MatrixSpace` base class.
- `Index Dim () const`
Accessor method for the dimension of the matrices in this matrix space.

Constructors/Destructors

- `SymMatrixSpace (Index dim)`
Constructor, given the dimension (identical to the number of rows and columns).
- virtual `~SymMatrixSpace ()`
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [SymMatrixSpace](#) ()
default constructor
- [SymMatrixSpace](#) (const [SymMatrixSpace](#) &)
- [SymMatrixSpace](#) & [operator=](#) (const [SymMatrixSpace](#) &)
Overloaded Equals Operator.

6.168.1 Detailed Description

[SymMatrixSpace](#) base class, corresponding to the [SymMatrix](#) base class.

Definition at line 81 of file IpSymMatrix.hpp.

6.168.2 Constructor & Destructor Documentation

6.168.2.1 Ipopt::SymMatrixSpace::SymMatrixSpace (Index *dim*) [inline]

Constructor, given the dimension (identical to the number of rows and columns).

Definition at line 89 of file IpSymMatrix.hpp.

6.168.2.2 virtual Ipopt::SymMatrixSpace::~SymMatrixSpace () [inline],[virtual]

Destructor.

Definition at line 95 of file IpSymMatrix.hpp.

6.168.2.3 Ipopt::SymMatrixSpace::SymMatrixSpace () [private]

default constructor

6.168.2.4 Ipopt::SymMatrixSpace::SymMatrixSpace (const SymMatrixSpace &) [private]

6.168.3 Member Function Documentation

6.168.3.1 virtual SymMatrix* Ipopt::SymMatrixSpace::MakeNewSymMatrix () const [pure virtual]

Pure virtual method for creating a new matrix of this specific type.

Implemented in [Ipopt::CompoundSymMatrixSpace](#), [Ipopt::SymTMatrixSpace](#), [Ipopt::LowRankUpdateSymMatrixSpace](#), [Ipopt::DenseSymMatrixSpace](#), [Ipopt::SymScaledMatrixSpace](#), [Ipopt::SumSymMatrixSpace](#), [Ipopt::IdentityMatrixSpace](#), [Ipopt::DiagMatrixSpace](#), and [Ipopt::ZeroSymMatrixSpace](#).

6.168.3.2 virtual Matrix* Ipopt::SymMatrixSpace::MakeNew () const [inline],[virtual]

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Reimplemented in [Ipopt::SymScaledMatrixSpace](#), and [Ipopt::ZeroSymMatrixSpace](#).

Definition at line 105 of file IpSymMatrix.hpp.

6.168.3.3 Index Ipopt::SymMatrixSpace::Dim () const [inline]

Accessor method for the dimension of the matrices in this matrix space.

Definition at line 113 of file IpSymMatrix.hpp.

6.168.3.4 SymMatrixSpace& Ipopt::SymMatrixSpace::operator= (const SymMatrixSpace &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

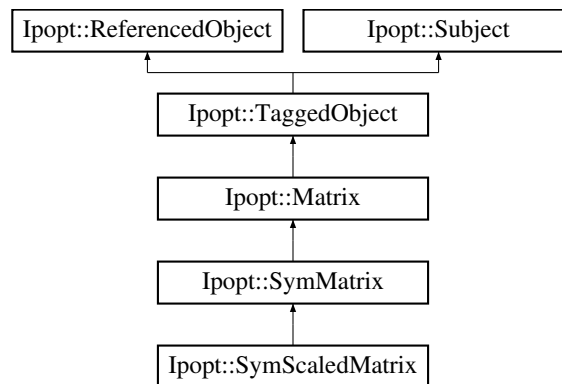
- [LinAlg/IpSymMatrix.hpp](#)

6.169 Ipopt::SymScaledMatrix Class Reference

Class for a [Matrix](#) in conjunction with its scaling factors for row and column scaling.

```
#include <IpSymScaledMatrix.hpp>
```

Inheritance diagram for Ipopt::SymScaledMatrix:



Public Member Functions

- void [SetUnscaledMatrix](#) (const [SmartPtr](#)< const [SymMatrix](#) > unscaled_matrix)
Set the unscaled matrix.
- void [SetUnscaledMatrixNonConst](#) (const [SmartPtr](#)< [SymMatrix](#) > &unscaled_matrix)
Set the unscaled matrix in a non-const version.
- [SmartPtr](#)< const [SymMatrix](#) > [GetUnscaledMatrix](#) () const
Return the unscaled matrix in const form.
- [SmartPtr](#)< [SymMatrix](#) > [GetUnscaledMatrixNonConst](#) ()
Return the unscaled matrix in non-const form.
- [SmartPtr](#)< const [Vector](#) > [RowColScaling](#) () const
return the vector for the row and column scaling

Constructors / Destructors

- [SymScaledMatrix](#) (const [SymScaledMatrixSpace](#) *owner_space)
Constructor, taking the owner_space.
- [~SymScaledMatrix](#) ()
Destructor.

Protected Member Functions

Methods overloaded from Matrix

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual bool [IsValidNumbersImpl](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [SymScaledMatrix](#) ()
Default Constructor.
- [SymScaledMatrix](#) (const [SymScaledMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [SymScaledMatrix](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< const [SymMatrix](#) > [matrix_](#)
const version of the unscaled matrix
- [SmartPtr](#)< [SymMatrix](#) > [nonconst_matrix_](#)
non-const version of the unscaled matrix
- [SmartPtr](#)< const [SymScaledMatrixSpace](#) > [owner_space_](#)
Matrix space stored as a [SymScaledMatrixSpace](#).

Additional Inherited Members

6.169.1 Detailed Description

Class for a [Matrix](#) in conjunction with its scaling factors for row and column scaling.

Operations on the matrix are performed using the scaled matrix. You can pull out the pointer to the unscaled matrix for unscaled calculations.

Definition at line 26 of file [IpSymScaledMatrix.hpp](#).

6.169.2 Constructor & Destructor Documentation

6.169.2.1 `Ipopt::SymScaledMatrix::SymScaledMatrix (const SymScaledMatrixSpace * owner_space)`

Constructor, taking the owner_space.

6.169.2.2 `Ipopt::SymScaledMatrix::~~SymScaledMatrix ()`

Destructor.

6.169.2.3 `Ipopt::SymScaledMatrix::SymScaledMatrix () [private]`

Default Constructor.

6.169.2.4 `Ipopt::SymScaledMatrix::SymScaledMatrix (const SymScaledMatrix &) [private]`

Copy Constructor.

6.169.3 Member Function Documentation

6.169.3.1 `void Ipopt::SymScaledMatrix::SetUnscaledMatrix (const SmartPtr< const SymMatrix > unscaled_matrix) [inline]`

Set the unscaled matrix.

Definition at line 194 of file `IpSymScaledMatrix.hpp`.

6.169.3.2 `void Ipopt::SymScaledMatrix::SetUnscaledMatrixNonConst (const SmartPtr< SymMatrix > & unscaled_matrix) [inline]`

Set the unscaled matrix in a non-const version.

Definition at line 202 of file `IpSymScaledMatrix.hpp`.

6.169.3.3 `SmartPtr< const SymMatrix > Ipopt::SymScaledMatrix::GetUnscaledMatrix () const [inline]`

Return the unscaled matrix in const form.

Definition at line 210 of file `IpSymScaledMatrix.hpp`.

6.169.3.4 `SmartPtr< SymMatrix > Ipopt::SymScaledMatrix::GetUnscaledMatrixNonConst () [inline]`

Return the unscaled matrix in non-const form.

Definition at line 216 of file `IpSymScaledMatrix.hpp`.

6.169.3.5 `SmartPtr< const Vector > Ipopt::SymScaledMatrix::RowColScaling () const [inline]`

return the vector for the row and column scaling

Definition at line 223 of file `IpSymScaledMatrix.hpp`.

6.169.3.6 `virtual void Ipopt::SymScaledMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.169.3.7 `virtual bool Ipopt::SymScaledMatrix::IsValidNumbersImpl () const` [protected],[virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

It is assumed here that the scaling factors are always valid numbers.

Reimplemented from [Ipopt::Matrix](#).

6.169.3.8 `virtual void Ipopt::SymScaledMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` [protected],[virtual]

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.169.3.9 `virtual void Ipopt::SymScaledMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected],[virtual]

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.169.3.10 `void Ipopt::SymScaledMatrix::operator= (const SymScaledMatrix &)` [private]

Overloaded Equals Operator.

6.169.4 Member Data Documentation

6.169.4.1 `SmartPtr<const SymMatrix> Ipopt::SymScaledMatrix::matrix_` [private]

const version of the unscaled matrix

Definition at line 97 of file IpSymScaledMatrix.hpp.

6.169.4.2 `SmartPtr<SymMatrix> Ipopt::SymScaledMatrix::nonconst_matrix_` [private]

non-const version of the unscaled matrix

Definition at line 99 of file IpSymScaledMatrix.hpp.

6.169.4.3 `SmartPtr<const SymScaledMatrixSpace> Ipopt::SymScaledMatrix::owner_space_` [private]

[Matrix](#) space stored as a [SymScaledMatrixSpace](#).

Definition at line 102 of file IpSymScaledMatrix.hpp.

The documentation for this class was generated from the following file:

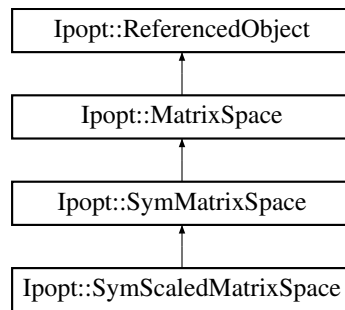
- [LinAlg/IpSymScaledMatrix.hpp](#)

6.170 Ipopt::SymScaledMatrixSpace Class Reference

This is the matrix space for [SymScaledMatrix](#).

```
#include <IpSymScaledMatrix.hpp>
```

Inheritance diagram for Ipopt::SymScaledMatrixSpace:



Public Member Functions

- [SymScaledMatrix](#) * [MakeNewSymScaledMatrix](#) (bool allocate_unscaled_matrix=false) const
Method for creating a new matrix of this specific type.
- virtual [SymMatrix](#) * [MakeNewSymMatrix](#) () const
Overloaded method from [SymMatrixSpace](#).
- virtual [Matrix](#) * [MakeNew](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.
- [SmartPtr](#)< const [Vector](#) > [RowColScaling](#) () const
return the vector for the row and column scaling
- [SmartPtr](#)< const [SymMatrixSpace](#) > [UnscaledMatrixSpace](#) () const
return the matrix space for the unscaled matrix

Constructors / Destructors

- [SymScaledMatrixSpace](#) (const [SmartPtr](#)< const [Vector](#) > &row_col_scaling, bool row_col_scaling_reciprocal, const [SmartPtr](#)< const [SymMatrixSpace](#) > &unscaled_matrix_space)
Constructor, given the number of row and columns blocks, as well as the total number of rows and columns.
- [~SymScaledMatrixSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [SymScaledMatrixSpace](#) ()
Default constructor.
- [SymScaledMatrixSpace](#) (const [SymScaledMatrixSpace](#) &)
Copy Constructor.
- [SymScaledMatrixSpace](#) & [operator=](#) (const [SymScaledMatrixSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [Vector](#) > [scaling_](#)
Row scaling vector.
- [SmartPtr](#)< const [SymMatrixSpace](#) > [unscaled_matrix_space_](#)
unscaled matrix space

6.170.1 Detailed Description

This is the matrix space for [SymScaledMatrix](#).

Definition at line 107 of file IpSymScaledMatrix.hpp.

6.170.2 Constructor & Destructor Documentation

6.170.2.1 `Ipopt::SymScaledMatrixSpace::SymScaledMatrixSpace (const SmartPtr< const Vector > & row_col_scaling, bool row_col_scaling_reciprocal, const SmartPtr< const SymMatrixSpace > & unscaled_matrix_space) [inline]`

Constructor, given the number of row and columns blocks, as well as the total number of rows and columns.

Definition at line 115 of file IpSymScaledMatrix.hpp.

6.170.2.2 `Ipopt::SymScaledMatrixSpace::~SymScaledMatrixSpace () [inline]`

Destructor.

Definition at line 129 of file IpSymScaledMatrix.hpp.

6.170.2.3 `Ipopt::SymScaledMatrixSpace::SymScaledMatrixSpace () [private]`

Default constructor.

6.170.2.4 `Ipopt::SymScaledMatrixSpace::SymScaledMatrixSpace (const SymScaledMatrixSpace &) [private]`

Copy Constructor.

6.170.3 Member Function Documentation

6.170.3.1 `SymScaledMatrix* Ipopt::SymScaledMatrixSpace::MakeNewSymScaledMatrix (bool allocate_unscaled_matrix = false) const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 134 of file IpSymScaledMatrix.hpp.

6.170.3.2 `virtual SymMatrix* Ipopt::SymScaledMatrixSpace::MakeNewSymMatrix () const [inline], [virtual]`

Overloaded method from [SymMatrixSpace](#).

Implements [Ipopt::SymMatrixSpace](#).

Definition at line 145 of file IpSymScaledMatrix.hpp.

6.170.3.3 `virtual Matrix* Ipopt::SymScaledMatrixSpace::MakeNew () const [inline], [virtual]`

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Reimplemented from [Ipopt::SymMatrixSpace](#).

Definition at line 151 of file IpSymScaledMatrix.hpp.

6.170.3.4 `SmartPtr<const Vector> Ipopt::SymScaledMatrixSpace::RowColScaling () const [inline]`

return the vector for the row and column scaling

Definition at line 157 of file IpSymScaledMatrix.hpp.

6.170.3.5 `SmartPtr<const SymMatrixSpace> Ipopt::SymScaledMatrixSpace::UnscaledMatrixSpace () const [inline]`

return the matrix space for the unscaled matrix

Definition at line 163 of file IpSymScaledMatrix.hpp.

6.170.3.6 `SymScaledMatrixSpace& Ipopt::SymScaledMatrixSpace::operator= (const SymScaledMatrixSpace &) [private]`

Overloaded Equals Operator.

6.170.4 Member Data Documentation

6.170.4.1 `SmartPtr<Vector> Ipopt::SymScaledMatrixSpace::scaling_ [private]`

Row scaling vector.

Definition at line 188 of file IpSymScaledMatrix.hpp.

6.170.4.2 `SmartPtr<const SymMatrixSpace> Ipopt::SymScaledMatrixSpace::unscaled_matrix_space_ [private]`

unscaled matrix space

Definition at line 190 of file IpSymScaledMatrix.hpp.

The documentation for this class was generated from the following file:

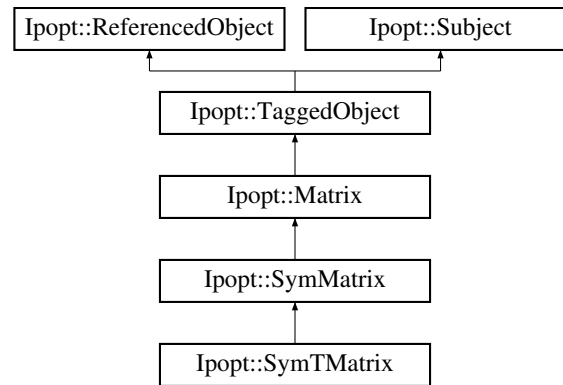
- [LinAlg/IpSymScaledMatrix.hpp](#)

6.171 Ipopt::SymTMatrix Class Reference

Class for symmetric matrices stored in triplet format.

```
#include <IpSymTMatrix.hpp>
```

Inheritance diagram for Ipopt::SymTMatrix:



Public Member Functions

Constructors / Destructors

- [SymTMatrix](#) (const [SymTMatrixSpace](#) *owner_space)
Constructor, taking the corresponding matrix space.
- [~SymTMatrix](#) ()
Destructor.

Changing the Values.

- void [SetValues](#) (const [Number](#) *Values)
Set values of nonzero elements.

Accessor Methods

- [Index Nonzeros](#) () const
Number of nonzero entries.
- const [Index](#) * [Irows](#) () const
Obtain pointer to the internal Index array irn_ without the intention to change the matrix data (USE WITH CARE!).
- const [Index](#) * [Jcols](#) () const
Obtain pointer to the internal Index array jcn_ without the intention to change the matrix data (USE WITH CARE!).
- [Number](#) * [Values](#) ()
Obtain pointer to the internal Number array values_ with the intention to change the matrix data (USE WITH CARE!).
- const [Number](#) * [Values](#) () const
Obtain pointer to the internal Number array values_ without the intention to change the matrix data (USE WITH CARE!).

Methods for providing copy of the matrix data

- void [FillStruct](#) ([ipfint](#) *Irn, [ipfint](#) *Jcn) const
Copy the nonzero structure into provided space.
- void [FillValues](#) ([Number](#) *Values) const
Copy the value data into provided space.

Protected Member Functions

Methods overloaded from matrix

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const

- *Matrix-vector multiply.*
- virtual bool `IsValidNumbersImpl` () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void `ComputeRowAMaxImpl` (Vector &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void `PrintImpl` (const Journalist &jnlst, EJournalLevel level, EJournalCategory category, const std::string &name, Index indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `SymTMatrix` ()
Default Constructor.
- `SymTMatrix` (const `SymTMatrix` &)
Copy Constructor.
- void `operator=` (const `SymTMatrix` &)
Overloaded Equals Operator.

Private Attributes

- const `SymTMatrixSpace` * `owner_space_`
Copy of the owner_space ptr as a `SymTMatrixSpace` instead of a `MatrixSpace`.
- `Number` * `values_`
Values of nonzeros.
- bool `initialized_`
Flag for Initialization.

Additional Inherited Members

6.171.1 Detailed Description

Class for symmetric matrices stored in triplet format.

In the triplet format, the nonzeros elements of a symmetric matrix is stored in three arrays, `lrm`, `jcn`, and `values`, all of length `Nonzeros`. The first two arrays indicate the location of a non-zero element (as the row and column indices), and the last array stores the value at that location. Off-diagonal elements need to be stored only once since the matrix is symmetric. For example, the element $a_{1,2} = a_{2,1}$ would be stored only once, either with `lrm[i]=1` and `jcn[i]=2`, or with `lrm[i]=2` and `jcn[i]=1`. Both representations are identical. If nonzero elements (or their symmetric counter part) are listed more than once, their values are added.

The structure of the nonzeros (i.e. the arrays `lrm` and `jcn`) cannot be changed after the matrix can be initialized. Only the values of the nonzero elements can be modified.

Note that the first row and column of a matrix has index 1, not 0.

Definition at line 42 of file `lpSymTMatrix.hpp`.

6.171.2 Constructor & Destructor Documentation

6.171.2.1 Ipopt::SymTMatrix::SymTMatrix (const SymTMatrixSpace * owner_space)

Constructor, taking the corresponding matrix space.

6.171.2.2 Ipopt::SymTMatrix::~~SymTMatrix ()

Destructor.

6.171.2.3 Ipopt::SymTMatrix::SymTMatrix () [private]

Default Constructor.

6.171.2.4 Ipopt::SymTMatrix::SymTMatrix (const SymTMatrix &) [private]

Copy Constructor.

6.171.3 Member Function Documentation

6.171.3.1 void Ipopt::SymTMatrix::SetValues (const Number * Values)

Set values of nonzero elements.

The values of the nonzero elements is copied from the incoming Number array. Important: It is assume that the order of the values in Values corresponds to the one of Irn and Jcn given to the matrix space.

6.171.3.2 Index Ipopt::SymTMatrix::Nonzeros () const [inline]

Number of nonzero entries.

Definition at line 234 of file IpSymTMatrix.hpp.

6.171.3.3 const Index * Ipopt::SymTMatrix::Irows () const [inline]

Obtain pointer to the internal Index array irn_ without the intention to change the matrix data (USE WITH CARE!).

This does not produce a copy, and lifetime is not guaranteed!

Definition at line 240 of file IpSymTMatrix.hpp.

6.171.3.4 const Index * Ipopt::SymTMatrix::Jcols () const [inline]

Obtain pointer to the internal Index array jcn_ without the intention to change the matrix data (USE WITH CARE!).

This does not produce a copy, and lifetime is not guaranteed!

Definition at line 246 of file IpSymTMatrix.hpp.

6.171.3.5 Number* Ipopt::SymTMatrix::Values ()

Obtain pointer to the internal Number array values_ with the intention to change the matrix data (USE WITH CARE!).

This does not produce a copy, and lifetime is not guaranteed!

6.171.3.6 const Number* Ipopt::SymTMatrix::Values () const

Obtain pointer to the internal Number array values_ without the intention to change the matrix data (USE WITH CARE!).

This does not produce a copy, and lifetime is not guaranteed!

6.171.3.7 `void Ipopt::SymTMatrix::FillStruct (ipfint * lrm, ipfint * Jcn) const`

Copy the nonzero structure into provided space.

6.171.3.8 `void Ipopt::SymTMatrix::FillValues (Number * Values) const`

Copy the value data into provided space.

6.171.3.9 `virtual void Ipopt::SymTMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const`
`[protected], [virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.171.3.10 `virtual bool Ipopt::SymTMatrix::IsValidNumbersImpl () const` `[protected], [virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

6.171.3.11 `virtual void Ipopt::SymTMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` `[protected], [virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in `rows_norms`. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

6.171.3.12 `virtual void Ipopt::SymTMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` `[protected], [virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.171.3.13 `void Ipopt::SymTMatrix::operator= (const SymTMatrix &)` `[private]`

Overloaded Equals Operator.

6.171.4 Member Data Documentation

6.171.4.1 `const SymTMatrixSpace* Ipopt::SymTMatrix::owner_space_` `[private]`

Copy of the `owner_space` ptr as a [SymTMatrixSpace](#) instead of a [MatrixSpace](#).

Definition at line 147 of file `IpSymTMatrix.hpp`.

6.171.4.2 `Number* Ipopt::SymTMatrix::values_` `[private]`

Values of nonzeros.

Definition at line 150 of file `IpSymTMatrix.hpp`.

6.171.4.3 bool Ipopt::SymTMatrix::initialized_ [private]

Flag for Initialization.

Definition at line 153 of file IpSymTMatrix.hpp.

The documentation for this class was generated from the following file:

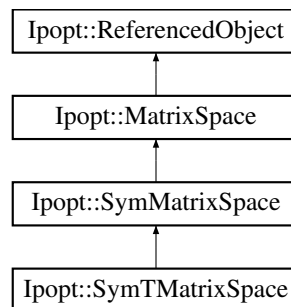
- [LinAlg/TMatrices/IpSymTMatrix.hpp](#)

6.172 Ipopt::SymTMatrixSpace Class Reference

This is the matrix space for a [SymTMatrix](#) with fixed sparsity structure.

```
#include <IpSymTMatrix.hpp>
```

Inheritance diagram for Ipopt::SymTMatrixSpace:



Public Member Functions

- virtual [SymMatrix](#) * [MakeNewSymMatrix](#) () const
Overloaded MakeNew method for the sYMMatrixSpace base class.
- [SymTMatrix](#) * [MakeNewSymTMatrix](#) () const
Method for creating a new matrix of this specific type.

Constructors / Destructors

- [SymTMatrixSpace](#) ([Index](#) dim, [Index](#) nonZeros, const [Index](#) *iRows, const [Index](#) *jCols)
Constructor, given the number of rows and columns (both as dim), as well as the number of nonzeros and the position of the nonzero elements.
- [~SymTMatrixSpace](#) ()
Destructor.

Methods describing Matrix structure

- [Index Nonzeros](#) () const
Number of non-zeros in the sparse matrix.
- const [Index](#) * [Irows](#) () const
Row index of each non-zero element.
- const [Index](#) * [Jcols](#) () const
Column index of each non-zero element.

Private Member Functions

Methods called by SymTMatrix for memory management

- `Number * AllocateInternalStorage () const`
Allocate internal storage for the [SymTMatrix](#) values.
- `void FreeInternalStorage (Number *values) const`
Deallocate internal storage for the [SymTMatrix](#) values.

Private Attributes

- `const Index nonZeros_`
- `Index * iRows_`
- `Index * jCols_`

Friends

- `class SymTMatrix`

6.172.1 Detailed Description

This is the matrix space for a [SymTMatrix](#) with fixed sparsity structure.

The sparsity structure is stored here in the matrix space.

Definition at line 161 of file `IpSymTMatrix.hpp`.

6.172.2 Constructor & Destructor Documentation

6.172.2.1 `Ipopt::SymTMatrixSpace::SymTMatrixSpace (Index dim, Index nonZeros, const Index * iRows, const Index * jCols)`

Constructor, given the number of rows and columns (both as dim), as well as the number of nonzeros and the position of the nonzero elements.

Note that the counting of the nonzeros starts a 1, i.e., `iRows[i]==1` and `jCols[i]==1` refers to the first element in the first row. This is in accordance with the HSL data structure. Off-diagonal elements are stored only once.

6.172.2.2 `Ipopt::SymTMatrixSpace::~SymTMatrixSpace ()`

Destructor.

6.172.3 Member Function Documentation

6.172.3.1 `virtual SymMatrix* Ipopt::SymTMatrixSpace::MakeNewSymMatrix () const [inline],[virtual]`

Overloaded MakeNew method for the `sYMMatrixSpace` base class.

Implements [Ipopt::SymMatrixSpace](#).

Definition at line 183 of file `IpSymTMatrix.hpp`.

6.172.3.2 `SymTMatrix* Ipopt::SymTMatrixSpace::MakeNewSymTMatrix () const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 189 of file `IpSymTMatrix.hpp`.

6.172.3.3 Index Ipopt::SymTMatrixSpace::Nonzeros () const [inline]

Number of non-zeros in the sparse matrix.

Definition at line 197 of file IpSymTMatrix.hpp.

6.172.3.4 const Index* Ipopt::SymTMatrixSpace::Irows () const [inline]

Row index of each non-zero element.

Definition at line 203 of file IpSymTMatrix.hpp.

6.172.3.5 const Index* Ipopt::SymTMatrixSpace::Jcols () const [inline]

Column index of each non-zero element.

Definition at line 209 of file IpSymTMatrix.hpp.

6.172.3.6 Number* Ipopt::SymTMatrixSpace::AllocateInternalStorage () const [private]

Allocate internal storage for the [SymTMatrix](#) values.

6.172.3.7 void Ipopt::SymTMatrixSpace::FreeInternalStorage (Number * values) const [private]

Deallocate internal storage for the [SymTMatrix](#) values.

6.172.4 Friends And Related Function Documentation**6.172.4.1** friend class SymTMatrix [friend]

Definition at line 229 of file IpSymTMatrix.hpp.

6.172.5 Member Data Documentation**6.172.5.1** const Index Ipopt::SymTMatrixSpace::nonZeros_ [private]

Definition at line 225 of file IpSymTMatrix.hpp.

6.172.5.2 Index* Ipopt::SymTMatrixSpace::iRows_ [private]

Definition at line 226 of file IpSymTMatrix.hpp.

6.172.5.3 Index* Ipopt::SymTMatrixSpace::jCols_ [private]

Definition at line 227 of file IpSymTMatrix.hpp.

The documentation for this class was generated from the following file:

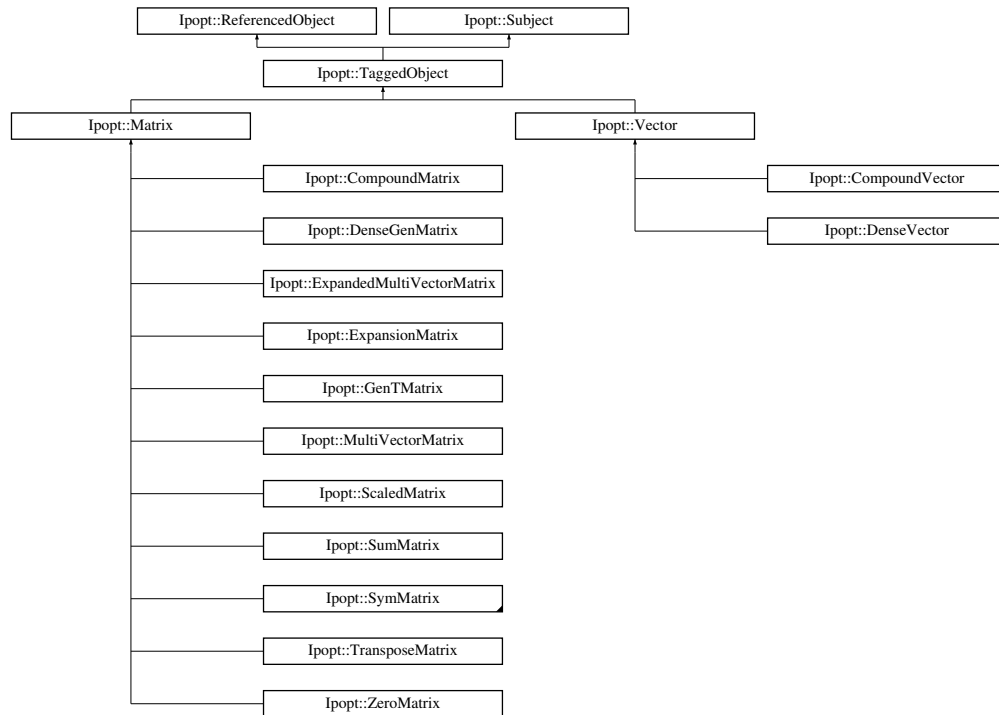
- LinAlg/TMatrices/[IpSymTMatrix.hpp](#)

6.173 Ipopt::TaggedObject Class Reference

[TaggedObject](#) class.

```
#include <IpTaggedObject.hpp>
```

Inheritance diagram for Ipopt::TaggedObject:



Public Types

- `typedef std::pair< const TaggedObject *, unsigned int > Tag`
Type for the Tag values.

Public Member Functions

- `TaggedObject ()`
Constructor.
- `virtual ~TaggedObject ()`
Destructor.
- `Tag GetTag () const`
Users of TaggedObjects call this to update their own internal tags every time they perform the expensive operation.
- `bool HasChanged (const Tag comparison_tag) const`
Users of TaggedObjects call this to check if the object HasChanged since they last updated their own internal tag.

Protected Member Functions

- `void ObjectChanged ()`
Objects derived from TaggedObject MUST call this method every time their internal state changes to update the internal tag for comparison.

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TaggedObject](#) (const [TaggedObject](#) &)
Copy Constructor.
- void [operator=](#) (const [TaggedObject](#) &)
Overloaded Equals Operator.

Private Attributes

- Tag::second_type [tagcount_](#)
The tag indicating the current state of the object.
- [Index](#) [cache_priority_](#)
The index indicating the cache priority for this [TaggedObject](#).

6.173.1 Detailed Description

[TaggedObject](#) class.

Often, certain calculations or operations are expensive, and it can be very inefficient to perform these calculations again if the input to the calculation has not changed since the result was last stored. This base class provides an efficient mechanism to update a tag, indicating that the object has changed. Users of a [TaggedObject](#) class, need their own Tag data member to keep track of the state of the [TaggedObject](#), the last time they performed a calculation. A basic use case for users of a class inheriting from [TaggedObject](#) follows like this:

1. Initialize your own Tag by its default constructor.
2. Before an expensive calculation, check if the [TaggedObject](#) has changed, passing in your own Tag, indicating the last time you used the object for the calculation. If it has changed, perform the calculation again, and store the result. If it has not changed, simply return the stored result.

Here is a simple example:

```
if (vector.HasChanged(my_vector_tag_)) {
    my_vector_tag_ = vector.GetTag();
    result = PerformExpensiveCalculation(vector);
    return result;
}
else {
    return result;
}
```

Objects derived from [TaggedObject](#) must indicate that they have changed to the base class using the protected member function [ObjectChanged\(\)](#). For example, a [Vector](#) class, inside its own set method, MUST call [ObjectChanged\(\)](#) to update the internally stored tag for comparison.

Definition at line 61 of file IpTaggedObject.hpp.

6.173.2 Member Typedef Documentation

6.173.2.1 `typedef std::pair<const TaggedObject*, unsigned int> Ipopt::TaggedObject::Tag`

Type for the Tag values.

To make the tag unique among all objects, we include the memory address of the object into the tag value.

Definition at line 70 of file `IpTaggedObject.hpp`.

6.173.3 Constructor & Destructor Documentation

6.173.3.1 `Ipopt::TaggedObject::TaggedObject () [inline]`

Constructor.

Definition at line 73 of file `IpTaggedObject.hpp`.

6.173.3.2 `virtual Ipopt::TaggedObject::~~TaggedObject () [inline],[virtual]`

Destructor.

Definition at line 84 of file `IpTaggedObject.hpp`.

6.173.3.3 `Ipopt::TaggedObject::TaggedObject (const TaggedObject &) [private]`

Copy Constructor.

6.173.4 Member Function Documentation

6.173.4.1 `Tag Ipopt::TaggedObject::GetTag () const [inline]`

Users of TaggedObjects call this to update their own internal tags every time they perform the expensive operation.

Definition at line 91 of file `IpTaggedObject.hpp`.

6.173.4.2 `bool Ipopt::TaggedObject::HasChanged (const Tag comparison_tag) const [inline]`

Users of TaggedObjects call this to check if the object HasChanged since they last updated their own internal tag.

Definition at line 101 of file `IpTaggedObject.hpp`.

6.173.4.3 `void Ipopt::TaggedObject::ObjectChanged () [inline],[protected]`

Objects derived from [TaggedObject](#) MUST call this method every time their internal state changes to update the internal tag for comparison.

Definition at line 110 of file `IpTaggedObject.hpp`.

6.173.4.4 `void Ipopt::TaggedObject::operator= (const TaggedObject &) [private]`

Overloaded Equals Operator.

6.173.5 Member Data Documentation

6.173.5.1 Tag::second_type Ipopt::TaggedObject::tagcount_ [private]

The tag indicating the current state of the object.

We use this to compare against the comparison_tag in the HasChanged method. This member is increased every time the object changes.

Definition at line 138 of file IpTaggedObject.hpp.

6.173.5.2 Index Ipopt::TaggedObject::cache_priority_ [private]

The index indicating the cache priority for this [TaggedObject](#).

If a result that depended on this [TaggedObject](#) is cached, it will be cached with this priority

Definition at line 145 of file IpTaggedObject.hpp.

The documentation for this class was generated from the following file:

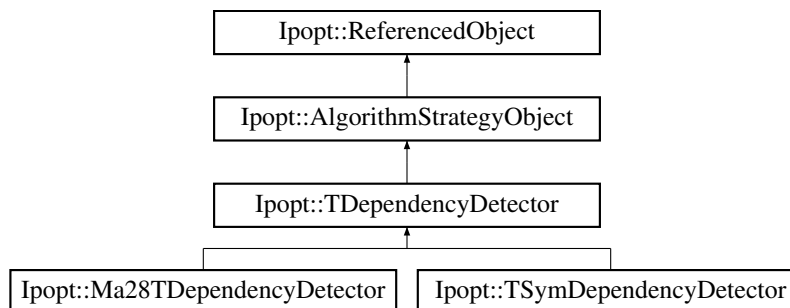
- [Common/IpTaggedObject.hpp](#)

6.174 Ipopt::TDependencyDetector Class Reference

Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

```
#include <IpTDependencyDetector.hpp>
```

Inheritance diagram for Ipopt::TDependencyDetector:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
Has to be called to initialize and reset these objects.
- virtual bool [DetermineDependentRows](#) ([Index](#) n_rows, [Index](#) n_cols, [Index](#) n_jac_nz, [Number](#) *jac_c_vals, [Index](#) *jac_c_iRow, [Index](#) *jac_c_jCol, std::list< [Index](#) > &c_deps)=0
Method determining the number of linearly dependent rows in the matrix and the indices of those rows.

Constructor/Destructor

- [TDependencyDetector](#) ()
- virtual [~TDependencyDetector](#) ()

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TDependencyDetector](#) (const [TDependencyDetector](#) &)
Copy Constructor.
- void [operator=](#) (const [TDependencyDetector](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.174.1 Detailed Description

Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

Definition at line 20 of file [IpTDependencyDetector.hpp](#).

6.174.2 Constructor & Destructor Documentation

6.174.2.1 [Ipopt::TDependencyDetector::TDependencyDetector \(\)](#) `[inline]`

Definition at line 25 of file [IpTDependencyDetector.hpp](#).

6.174.2.2 [virtual Ipopt::TDependencyDetector::~~TDependencyDetector \(\)](#) `[inline],[virtual]`

Definition at line 28 of file [IpTDependencyDetector.hpp](#).

6.174.2.3 [Ipopt::TDependencyDetector::TDependencyDetector \(const TDependencyDetector & \)](#) `[private]`

Copy Constructor.

6.174.3 Member Function Documentation

6.174.3.1 [virtual bool Ipopt::TDependencyDetector::InitializeImpl \(const OptionsList & options, const std::string & prefix \)](#) `[pure virtual]`

Has to be called to initialize and reset these objects.

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::TSymDependencyDetector](#), and [Ipopt::Ma28TDependencyDetector](#).

6.174.3.2 [virtual bool Ipopt::TDependencyDetector::DetermineDependentRows \(Index n_rows, Index n_cols, Index n_jac_nz, Number * jac_c_vals, Index * jac_c_iRow, Index * jac_c_jCol, std::list< Index > & c_deps \)](#) `[pure virtual]`

Method determining the number of linearly dependent rows in the matrix and the indices of those rows.

We assume that the matrix is available in "Triplet" format (MA28 format), and that the arrays given to this method can be modified internally, i.e., they are not used by the calling program anymore after this call. This method returns false if there was a problem with the underlying linear solver.

Implemented in [Ipopt::TSymDependencyDetector](#), and [Ipopt::Ma28TDependencyDetector](#).

6.174.3.3 void Ipopt::TDependencyDetector::operator= (const TDependencyDetector &) [private]

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

- Algorithm/LinearSolvers/[IpTDependencyDetector.hpp](#)

6.175 Ipopt::TimedTask Class Reference

This class is used to collect timing information for a particular task.

```
#include <IpTimedTask.hpp>
```

Public Member Functions

- void [Reset](#) ()
Method for resetting time to zero.
- void [Start](#) ()
Method that is called before execution of the task.
- void [End](#) ()
Method that is called after execution of the task.
- void [EndIfStarted](#) ()
Method that is called after execution of the task for which timing might have been started.
- [Number TotalCpuTime](#) () const
Method returning total CPU time spend for task so far.
- [Number TotalSysTime](#) () const
Method returning total system time spend for task so far.
- [Number TotalWallclockTime](#) () const
Method returning total wall clock time spend for task so far.

Constructors/Destructors

- [TimedTask](#) ()
Default constructor.
- [~TimedTask](#) ()
Default destructor.

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TimedTask](#) (const [TimedTask](#) &)
Copy Constructor.
- void [operator=](#) (const [TimedTask](#) &)
Overloaded Equals Operator.

Private Attributes

- [Number start_cpu_time_](#)
CPU time at beginning of task.
- [Number total_cpu_time_](#)
Total CPU time for task measured so far.
- [Number start_sys_time_](#)
System time at beginning of task.
- [Number total_sys_time_](#)
Total system time for task measured so far.
- [Number start_wall_time_](#)
Wall clock time at beginning of task.
- [Number total_wall_time_](#)
Total wall clock time for task measured so far.

fields for debugging

- bool [start_called_](#)
- bool [end_called_](#)

6.175.1 Detailed Description

This class is used to collect timing information for a particular task.

Definition at line 18 of file IpTimedTask.hpp.

6.175.2 Constructor & Destructor Documentation

6.175.2.1 Ipopt::TimedTask::TimedTask () [inline]

Default constructor.

Definition at line 24 of file IpTimedTask.hpp.

6.175.2.2 Ipopt::TimedTask::~~TimedTask () [inline]

Default destructor.

Definition at line 34 of file IpTimedTask.hpp.

6.175.2.3 Ipopt::TimedTask::TimedTask (const TimedTask &) [private]

Copy Constructor.

6.175.3 Member Function Documentation

6.175.3.1 void Ipopt::TimedTask::Reset () [inline]

Method for resetting time to zero.

Definition at line 39 of file IpTimedTask.hpp.

6.175.3.2 void Ipopt::TimedTask::Start () [inline]

Method that is called before execution of the task.

Definition at line 49 of file IpTimedTask.hpp.

6.175.3.3 void Ipopt::TimedTask::End () [inline]

Method that is called after execution of the task.

Definition at line 61 of file IpTimedTask.hpp.

6.175.3.4 void Ipopt::TimedTask::EndIfStarted () [inline]

Method that is called after execution of the task for which timing might have been started.

This only updates the timing if the timing has indeed been conducted. This is useful to stop timing after catching exceptions.

Definition at line 76 of file IpTimedTask.hpp.

6.175.3.5 Number Ipopt::TimedTask::TotalCpuTime () const [inline]

Method returning total CPU time spend for task so far.

Definition at line 89 of file IpTimedTask.hpp.

6.175.3.6 Number Ipopt::TimedTask::TotalSysTime () const [inline]

Method returning total system time spend for task so far.

Definition at line 96 of file IpTimedTask.hpp.

6.175.3.7 Number Ipopt::TimedTask::TotalWallclockTime () const [inline]

Method returning total wall clock time spend for task so far.

Definition at line 103 of file IpTimedTask.hpp.

6.175.3.8 void Ipopt::TimedTask::operator= (const TimedTask &) [private]

Overloaded Equals Operator.

6.175.4 Member Data Documentation**6.175.4.1 Number Ipopt::TimedTask::start_cputime_ [private]**

CPU time at beginning of task.

Definition at line 125 of file IpTimedTask.hpp.

6.175.4.2 Number Ipopt::TimedTask::total_cputime_ [private]

Total CPU time for task measured so far.

Definition at line 127 of file IpTimedTask.hpp.

6.175.4.3 Number Ipopt::TimedTask::start_systime_ [private]

System time at beginning of task.

Definition at line 129 of file IpTimedTask.hpp.

6.175.4.4 **Number** Ipopt::TimedTask::total_systime_ [private]

Total system time for task measured so far.

Definition at line 131 of file IpTimedTask.hpp.

6.175.4.5 **Number** Ipopt::TimedTask::start_waltime_ [private]

Wall clock time at beginning of task.

Definition at line 133 of file IpTimedTask.hpp.

6.175.4.6 **Number** Ipopt::TimedTask::total_waltime_ [private]

Total wall clock time for task measured so far.

Definition at line 135 of file IpTimedTask.hpp.

6.175.4.7 **bool** Ipopt::TimedTask::start_called_ [private]

Definition at line 139 of file IpTimedTask.hpp.

6.175.4.8 **bool** Ipopt::TimedTask::end_called_ [private]

Definition at line 140 of file IpTimedTask.hpp.

The documentation for this class was generated from the following file:

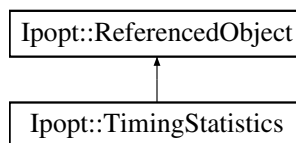
- [Common/IpTimedTask.hpp](#)

6.176 Ipopt::TimingStatistics Class Reference

This class collects all timing statistics for [Ipopt](#).

```
#include <IpTimingStatistics.hpp>
```

Inheritance diagram for Ipopt::TimingStatistics:



Public Member Functions

- void [ResetTimes](#) ()
Method for resetting all times.
- void [PrintAllTimingStatistics](#) ([Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category) const
Method for printing all timing information.

Constructors/Destructors

- [TimingStatistics](#) ()

Default constructor.

- virtual [~TimingStatistics](#) ()

Default destructor.

Accessor methods to all timed tasks.

- [TimedTask](#) & [OverallAlgorithm](#) ()
- [TimedTask](#) & [PrintProblemStatistics](#) ()
- [TimedTask](#) & [InitializeIterates](#) ()
- [TimedTask](#) & [UpdateHessian](#) ()
- [TimedTask](#) & [OutputIteration](#) ()
- [TimedTask](#) & [UpdateBarrierParameter](#) ()
- [TimedTask](#) & [ComputeSearchDirection](#) ()
- [TimedTask](#) & [ComputeAcceptableTrialPoint](#) ()
- [TimedTask](#) & [AcceptTrialPoint](#) ()
- [TimedTask](#) & [CheckConvergence](#) ()
- [TimedTask](#) & [PDSystemSolverTotal](#) ()
- [TimedTask](#) & [PDSystemSolverSolveOnce](#) ()
- [TimedTask](#) & [ComputeResiduals](#) ()
- [TimedTask](#) & [StdAugSystemSolverMultiSolve](#) ()
- [TimedTask](#) & [LinearSystemScaling](#) ()
- [TimedTask](#) & [LinearSystemSymbolicFactorization](#) ()
- [TimedTask](#) & [LinearSystemFactorization](#) ()
- [TimedTask](#) & [LinearSystemBackSolve](#) ()
- [TimedTask](#) & [LinearSystemStructureConverter](#) ()
- [TimedTask](#) & [LinearSystemStructureConverterInit](#) ()
- [TimedTask](#) & [QualityFunctionSearch](#) ()
- [TimedTask](#) & [TryCorrector](#) ()
- [TimedTask](#) & [Task1](#) ()
- [TimedTask](#) & [Task2](#) ()
- [TimedTask](#) & [Task3](#) ()
- [TimedTask](#) & [Task4](#) ()
- [TimedTask](#) & [Task5](#) ()
- [TimedTask](#) & [Task6](#) ()

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TimingStatistics](#) (const [TimingStatistics](#) &)
 - void [operator=](#) (const [TimingStatistics](#) &)
- Copy Constructor.*
Overloaded Equals Operator.

Private Attributes

All timed tasks.

- [TimedTask](#) [OverallAlgorithm_](#)
- [TimedTask](#) [PrintProblemStatistics_](#)
- [TimedTask](#) [InitializeIterates_](#)
- [TimedTask](#) [UpdateHessian_](#)
- [TimedTask](#) [OutputIteration_](#)

- [TimedTask UpdateBarrierParameter_](#)
- [TimedTask ComputeSearchDirection_](#)
- [TimedTask ComputeAcceptableTrialPoint_](#)
- [TimedTask AcceptTrialPoint_](#)
- [TimedTask CheckConvergence_](#)
- [TimedTask PDSolverTotal_](#)
- [TimedTask PDSolverSolveOnce_](#)
- [TimedTask ComputeResiduals_](#)
- [TimedTask StdAugSystemSolverMultiSolve_](#)
- [TimedTask LinearSystemScaling_](#)
- [TimedTask LinearSystemSymbolicFactorization_](#)
- [TimedTask LinearSystemFactorization_](#)
- [TimedTask LinearSystemBackSolve_](#)
- [TimedTask LinearSystemStructureConverter_](#)
- [TimedTask LinearSystemStructureConverterInit_](#)
- [TimedTask QualityFunctionSearch_](#)
- [TimedTask TryCorrector_](#)
- [TimedTask Task1_](#)
- [TimedTask Task2_](#)
- [TimedTask Task3_](#)
- [TimedTask Task4_](#)
- [TimedTask Task5_](#)
- [TimedTask Task6_](#)

6.176.1 Detailed Description

This class collects all timing statistics for [Ipopt](#).

Definition at line 20 of file IpTimingStatistics.hpp.

6.176.2 Constructor & Destructor Documentation

6.176.2.1 `Ipopt::TimingStatistics::TimingStatistics () [inline]`

Default constructor.

Definition at line 26 of file IpTimingStatistics.hpp.

6.176.2.2 `virtual Ipopt::TimingStatistics::~~TimingStatistics () [inline],[virtual]`

Default destructor.

Definition at line 30 of file IpTimingStatistics.hpp.

6.176.2.3 `Ipopt::TimingStatistics::TimingStatistics (const TimingStatistics &) [private]`

Copy Constructor.

6.176.3 Member Function Documentation

6.176.3.1 `void Ipopt::TimingStatistics::ResetTimes ()`

Method for resetting all times.

6.176.3.2 `void Ipopt::TimingStatistics::PrintAllTimingStatistics (Journalist & jnlst, EJournalLevel level, EJournalCategory category) const`

Method for printing all timing information.

6.176.3.3 `TimedTask& Ipopt::TimingStatistics::OverallAlgorithm () [inline]`

Definition at line 44 of file IpTimingStatistics.hpp.

6.176.3.4 `TimedTask& Ipopt::TimingStatistics::PrintProblemStatistics () [inline]`

Definition at line 48 of file IpTimingStatistics.hpp.

6.176.3.5 `TimedTask& Ipopt::TimingStatistics::InitializeIterates () [inline]`

Definition at line 52 of file IpTimingStatistics.hpp.

6.176.3.6 `TimedTask& Ipopt::TimingStatistics::UpdateHessian () [inline]`

Definition at line 56 of file IpTimingStatistics.hpp.

6.176.3.7 `TimedTask& Ipopt::TimingStatistics::OutputIteration () [inline]`

Definition at line 60 of file IpTimingStatistics.hpp.

6.176.3.8 `TimedTask& Ipopt::TimingStatistics::UpdateBarrierParameter () [inline]`

Definition at line 64 of file IpTimingStatistics.hpp.

6.176.3.9 `TimedTask& Ipopt::TimingStatistics::ComputeSearchDirection () [inline]`

Definition at line 68 of file IpTimingStatistics.hpp.

6.176.3.10 `TimedTask& Ipopt::TimingStatistics::ComputeAcceptableTrialPoint () [inline]`

Definition at line 72 of file IpTimingStatistics.hpp.

6.176.3.11 `TimedTask& Ipopt::TimingStatistics::AcceptTrialPoint () [inline]`

Definition at line 76 of file IpTimingStatistics.hpp.

6.176.3.12 `TimedTask& Ipopt::TimingStatistics::CheckConvergence () [inline]`

Definition at line 80 of file IpTimingStatistics.hpp.

6.176.3.13 `TimedTask& Ipopt::TimingStatistics::PDSolverSystemTotal () [inline]`

Definition at line 85 of file IpTimingStatistics.hpp.

6.176.3.14 `TimedTask& Ipopt::TimingStatistics::PDSolverSystemSolveOnce () [inline]`

Definition at line 89 of file IpTimingStatistics.hpp.

6.176.3.15 `TimedTask& Ipopt::TimingStatistics::ComputeResiduals () [inline]`

Definition at line 93 of file IpTimingStatistics.hpp.

6.176.3.16 `TimedTask& Ipopt::TimingStatistics::StdAugSystemSolverMultiSolve () [inline]`

Definition at line 97 of file IpTimingStatistics.hpp.

6.176.3.17 `TimedTask& Ipopt::TimingStatistics::LinearSystemScaling () [inline]`

Definition at line 101 of file IpTimingStatistics.hpp.

6.176.3.18 `TimedTask& Ipopt::TimingStatistics::LinearSystemSymbolicFactorization () [inline]`

Definition at line 105 of file IpTimingStatistics.hpp.

6.176.3.19 `TimedTask& Ipopt::TimingStatistics::LinearSystemFactorization () [inline]`

Definition at line 109 of file IpTimingStatistics.hpp.

6.176.3.20 `TimedTask& Ipopt::TimingStatistics::LinearSystemBackSolve () [inline]`

Definition at line 113 of file IpTimingStatistics.hpp.

6.176.3.21 `TimedTask& Ipopt::TimingStatistics::LinearSystemStructureConverter () [inline]`

Definition at line 117 of file IpTimingStatistics.hpp.

6.176.3.22 `TimedTask& Ipopt::TimingStatistics::LinearSystemStructureConverterInit () [inline]`

Definition at line 121 of file IpTimingStatistics.hpp.

6.176.3.23 `TimedTask& Ipopt::TimingStatistics::QualityFunctionSearch () [inline]`

Definition at line 125 of file IpTimingStatistics.hpp.

6.176.3.24 `TimedTask& Ipopt::TimingStatistics::TryCorrector () [inline]`

Definition at line 129 of file IpTimingStatistics.hpp.

6.176.3.25 `TimedTask& Ipopt::TimingStatistics::Task1 () [inline]`

Definition at line 134 of file IpTimingStatistics.hpp.

6.176.3.26 `TimedTask& Ipopt::TimingStatistics::Task2 () [inline]`

Definition at line 138 of file IpTimingStatistics.hpp.

6.176.3.27 `TimedTask& Ipopt::TimingStatistics::Task3 () [inline]`

Definition at line 142 of file IpTimingStatistics.hpp.

6.176.3.28 `TimedTask& Ipopt::TimingStatistics::Task4 () [inline]`

Definition at line 146 of file IpTimingStatistics.hpp.

6.176.3.29 `TimedTask& Ipopt::TimingStatistics::Task5 () [inline]`

Definition at line 150 of file IpTimingStatistics.hpp.

6.176.3.30 `TimedTask& Ipopt::TimingStatistics::Task6 () [inline]`

Definition at line 154 of file IpTimingStatistics.hpp.

6.176.3.31 `void Ipopt::TimingStatistics::operator=(const TimingStatistics &)` [private]

Overloaded Equals Operator.

6.176.4 Member Data Documentation

6.176.4.1 `TimedTask Ipopt::TimingStatistics::OverallAlgorithm_` [private]

Definition at line 178 of file IpTimingStatistics.hpp.

6.176.4.2 `TimedTask Ipopt::TimingStatistics::PrintProblemStatistics_` [private]

Definition at line 179 of file IpTimingStatistics.hpp.

6.176.4.3 `TimedTask Ipopt::TimingStatistics::Initializelterates_` [private]

Definition at line 180 of file IpTimingStatistics.hpp.

6.176.4.4 `TimedTask Ipopt::TimingStatistics::UpdateHessian_` [private]

Definition at line 181 of file IpTimingStatistics.hpp.

6.176.4.5 `TimedTask Ipopt::TimingStatistics::OutputIteration_` [private]

Definition at line 182 of file IpTimingStatistics.hpp.

6.176.4.6 `TimedTask Ipopt::TimingStatistics::UpdateBarrierParameter_` [private]

Definition at line 183 of file IpTimingStatistics.hpp.

6.176.4.7 `TimedTask Ipopt::TimingStatistics::ComputeSearchDirection_` [private]

Definition at line 184 of file IpTimingStatistics.hpp.

6.176.4.8 `TimedTask Ipopt::TimingStatistics::ComputeAcceptableTrialPoint_` [private]

Definition at line 185 of file IpTimingStatistics.hpp.

6.176.4.9 `TimedTask Ipopt::TimingStatistics::AcceptTrialPoint_` [private]

Definition at line 186 of file IpTimingStatistics.hpp.

6.176.4.10 `TimedTask Ipopt::TimingStatistics::CheckConvergence_` [private]

Definition at line 187 of file IpTimingStatistics.hpp.

6.176.4.11 `TimedTask Ipopt::TimingStatistics::PDSolverTotal_` [private]

Definition at line 189 of file IpTimingStatistics.hpp.

6.176.4.12 `TimedTask Ipopt::TimingStatistics::PDSolverSolveOnce_` [private]

Definition at line 190 of file IpTimingStatistics.hpp.

6.176.4.13 `TimedTask Ipopt::TimingStatistics::ComputeResiduals_` [private]

Definition at line 191 of file IpTimingStatistics.hpp.

6.176.4.14 **TimedTask Ipopt::TimingStatistics::StdAugSystemSolverMultiSolve_** [private]

Definition at line 192 of file IpTimingStatistics.hpp.

6.176.4.15 **TimedTask Ipopt::TimingStatistics::LinearSystemScaling_** [private]

Definition at line 193 of file IpTimingStatistics.hpp.

6.176.4.16 **TimedTask Ipopt::TimingStatistics::LinearSystemSymbolicFactorization_** [private]

Definition at line 194 of file IpTimingStatistics.hpp.

6.176.4.17 **TimedTask Ipopt::TimingStatistics::LinearSystemFactorization_** [private]

Definition at line 195 of file IpTimingStatistics.hpp.

6.176.4.18 **TimedTask Ipopt::TimingStatistics::LinearSystemBackSolve_** [private]

Definition at line 196 of file IpTimingStatistics.hpp.

6.176.4.19 **TimedTask Ipopt::TimingStatistics::LinearSystemStructureConverter_** [private]

Definition at line 197 of file IpTimingStatistics.hpp.

6.176.4.20 **TimedTask Ipopt::TimingStatistics::LinearSystemStructureConverterInit_** [private]

Definition at line 198 of file IpTimingStatistics.hpp.

6.176.4.21 **TimedTask Ipopt::TimingStatistics::QualityFunctionSearch_** [private]

Definition at line 199 of file IpTimingStatistics.hpp.

6.176.4.22 **TimedTask Ipopt::TimingStatistics::TryCorrector_** [private]

Definition at line 200 of file IpTimingStatistics.hpp.

6.176.4.23 **TimedTask Ipopt::TimingStatistics::Task1_** [private]

Definition at line 202 of file IpTimingStatistics.hpp.

6.176.4.24 **TimedTask Ipopt::TimingStatistics::Task2_** [private]

Definition at line 203 of file IpTimingStatistics.hpp.

6.176.4.25 **TimedTask Ipopt::TimingStatistics::Task3_** [private]

Definition at line 204 of file IpTimingStatistics.hpp.

6.176.4.26 **TimedTask Ipopt::TimingStatistics::Task4_** [private]

Definition at line 205 of file IpTimingStatistics.hpp.

6.176.4.27 **TimedTask Ipopt::TimingStatistics::Task5_** [private]

Definition at line 206 of file IpTimingStatistics.hpp.

6.176.4.28 TimedTask Ipopt::TimingStatistics::Task6_ [private]

Definition at line 207 of file IpTimingStatistics.hpp.

The documentation for this class was generated from the following file:

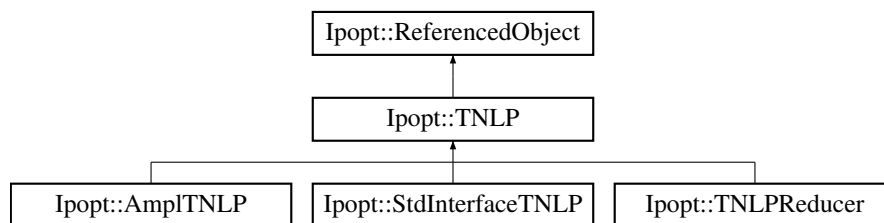
- [Algorithm/IpTimingStatistics.hpp](#)

6.177 Ipopt::TNLP Class Reference

Base class for all [NLP](#)'s that use standard triplet matrix form and dense vectors.

```
#include <IpTNLP.hpp>
```

Inheritance diagram for Ipopt::TNLP:



Public Types

- enum [LinearityType](#) { [LINEAR](#), [NON_LINEAR](#) }
- Type of the constraints.*

Public Member Functions

- [DECLARE_STD_EXCEPTION](#) (INVALID_TNLP)

Constructors/Destructors

- [TNLP](#) ()
 - virtual [~TNLP](#) ()
- Default destructor.*

Solution Methods

- virtual void [finalize_solution](#) (SolverReturn status, [Index](#) n, const [Number](#) *x, const [Number](#) *z_L, const [Number](#) *z_U, [Index](#) m, const [Number](#) *g, const [Number](#) *lambda, [Number](#) obj_value, const [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)=0
- This method is called when the algorithm is complete so the [TNLP](#) can store/write the solution.*
- virtual void [finalize_metadata](#) ([Index](#) n, const [StringMetaDataMapType](#) &var_string_md, const [IntegerMetaDataMapType](#) &var_integer_md, const [NumericMetaDataMapType](#) &var_numeric_md, [Index](#) m, const [StringMetaDataMapType](#) &con_string_md, const [IntegerMetaDataMapType](#) &con_integer_md, const [NumericMetaDataMapType](#) &con_numeric_md)
- This method is called just before finalize_solution.*
- virtual bool [intermediate_callback](#) ([AlgorithmMode](#) mode, [Index](#) iter, [Number](#) obj_value, [Number](#) inf_pr, [Number](#) inf_du, [Number](#) mu, [Number](#) d_norm, [Number](#) regularization_size, [Number](#) alpha_du, [Number](#) alpha_pr, [Index](#) ls_trials, const [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)

Intermediate Callback method for the user.

Methods for quasi-Newton approximation. If the second

derivatives are approximated by *lpopt*, it is better to do this only in the space of nonlinear variables.

The following methods are call by *lpopt* if the quasi-Newton approximation is selected. If -1 is returned as number of nonlinear variables, *lpopt* assumes that all variables are nonlinear. Otherwise, it calls *get_list_of_nonlinear_variables* with an array into which the indices of the nonlinear variables should be written - the array has the lengths *num_nonlin_vars*, which is identical with the return value of *get_number_of_nonlinear_variables()*. It is assumed that the indices are counted starting with 1 in the FORTRAN_STYLE, and 0 for the C_STYLE.

- virtual [Index](#) *get_number_of_nonlinear_variables* ()
- virtual bool *get_list_of_nonlinear_variables* ([Index](#) num_nonlin_vars, [Index](#) *pos_nonlin_vars)

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TNLP](#) (const [TNLP](#) &)
Default Constructor.
- void *operator=* (const [TNLP](#) &)
Overloaded Equals Operator.

methods to gather information about the NLP

- enum [IndexStyleEnum](#) { [C_STYLE](#) =0, [FORTRAN_STYLE](#) =1 }
overload this method to return the number of variables and constraints, and the number of non-zeros in the jacobian and the hessian.
- typedef std::map< std::string, std::vector< std::string > > [StringMetaDataMapType](#)
- typedef std::map< std::string, std::vector< [Index](#) > > [IntegerMetaDataMapType](#)
- typedef std::map< std::string, std::vector< [Number](#) > > [NumericMetaDataMapType](#)
- virtual bool *get_nlp_info* ([Index](#) &n, [Index](#) &m, [Index](#) &nnz_jac_g, [Index](#) &nnz_h_lag, [IndexStyleEnum](#) &index_style)=0
- virtual bool *get_var_con_metadata* ([Index](#) n, [StringMetaDataMapType](#) &var_string_md, [IntegerMetaDataMapType](#) &var_integer_md, [NumericMetaDataMapType](#) &var_numeric_md, [Index](#) m, [StringMetaDataMapType](#) &con_string_md, [IntegerMetaDataMapType](#) &con_integer_md, [NumericMetaDataMapType](#) &con_numeric_md)
overload this method to return any meta data for the variables and the constraints
- virtual bool *get_bounds_info* ([Index](#) n, [Number](#) *x_l, [Number](#) *x_u, [Index](#) m, [Number](#) *g_l, [Number](#) *g_u)=0
overload this method to return the information about the bound on the variables and constraints.
- virtual bool *get_scaling_parameters* ([Number](#) &obj_scaling, bool &use_x_scaling, [Index](#) n, [Number](#) *x_scaling, bool &use_g_scaling, [Index](#) m, [Number](#) *g_scaling)
overload this method to return scaling parameters.
- virtual bool *get_variables_linearity* ([Index](#) n, [LinearityType](#) *var_types)
overload this method to return the variables linearity (TNLP::LINEAR or TNLP::NON_LINEAR).
- virtual bool *get_constraints_linearity* ([Index](#) m, [LinearityType](#) *const_types)

overload this method to return the constraint linearity.

- virtual bool `get_starting_point` (Index n, bool init_x, Number *x, bool init_z, Number *z_L, Number *z_U, Index m, bool init_lambda, Number *lambda)=0

overload this method to return the starting point.

- virtual bool `get_warm_start_iterate` (IteratesVector &warm_start_iterate)

overload this method to provide an Ipopt iterate (already in the form Ipopt requires it internally) for a warm start.

- virtual bool `eval_f` (Index n, const Number *x, bool new_x, Number &obj_value)=0

overload this method to return the value of the objective function

- virtual bool `eval_grad_f` (Index n, const Number *x, bool new_x, Number *grad_f)=0

overload this method to return the vector of the gradient of the objective w.r.t.

- virtual bool `eval_g` (Index n, const Number *x, bool new_x, Index m, Number *g)=0

overload this method to return the vector of constraint values

- virtual bool `eval_jac_g` (Index n, const Number *x, bool new_x, Index m, Index nele_jac, Index *iRow, Index *jCol, Number *values)=0

overload this method to return the jacobian of the constraints.

- virtual bool `eval_h` (Index n, const Number *x, bool new_x, Number obj_factor, Index m, const Number *lambda, bool new_lambda, Index nele_hess, Index *iRow, Index *jCol, Number *values)

overload this method to return the hessian of the lagrangian.

6.177.1 Detailed Description

Base class for all NLP's that use standard triplet matrix form and dense vectors.

This is the standard base class for all NLP's that use the standard triplet matrix form (as for Harwell routines) and dense vectors. The class `TNLPAdapter` then converts this interface to an interface that can be used directly by ipopt.

This interface presents the problem form:

$\min f(x)$

s.t. $gL \leq g(x) \leq gU$

$xL \leq x \leq xU$

In order to specify an equality constraint, set $gL_i = gU_i = \text{rhs}$. The value that indicates "infinity" for the bounds (i.e. the variable or constraint has no lower bound (-infinity) or upper bound (+infinity)) is set through the option `nlp_lower_bound_inf` and `nlp_upper_bound_inf`. To indicate that a variable has no upper or lower bound, set the bound to `-ipopt_inf` or `+ipopt_inf` respectively

Definition at line 50 of file `IpTNLP.hpp`.

6.177.2 Member Typedef Documentation

6.177.2.1 `typedef std::map<std::string, std::vector<std::string> > Ipopt::TNLP::StringMetaDataMapType`

Definition at line 84 of file `IpTNLP.hpp`.

6.177.2.2 `typedef std::map<std::string, std::vector<Index> > Ipopt::TNLP::IntegerMetaDataMapType`

Definition at line 85 of file `IpTNLP.hpp`.

6.177.2.3 `typedef std::map<std::string, std::vector<Number> > Ipopt::TNLP::NumericMetaDataMapType`

Definition at line 86 of file `IpTNLP.hpp`.

6.177.3 Member Enumeration Documentation

6.177.3.1 enum Ipopt::TNLP::LinearityType

Type of the constraints.

Enumerator

LINEAR Constraint/Variable is linear.

NON_LINEAR Constraint/Variable is non-linear.

Definition at line 54 of file IpTNLP.hpp.

6.177.3.2 enum Ipopt::TNLP::IndexStyleEnum

overload this method to return the number of variables and constraints, and the number of non-zeros in the jacobian and the hessian.

The index_style parameter lets you specify C or Fortran style indexing for the sparse matrix iRow and jCol parameters. C_STYLE is 0-based, and FORTRAN_STYLE is 1-based.

Enumerator

C_STYLE

FORTRAN_STYLE

Definition at line 80 of file IpTNLP.hpp.

6.177.4 Constructor & Destructor Documentation

6.177.4.1 Ipopt::TNLP::TNLP () [inline]

Definition at line 62 of file IpTNLP.hpp.

6.177.4.2 virtual Ipopt::TNLP::~~TNLP () [inline],[virtual]

Default destructor.

Definition at line 66 of file IpTNLP.hpp.

6.177.4.3 Ipopt::TNLP::TNLP (const TNLP &) [private]

Default Constructor.

Copy Constructor

6.177.5 Member Function Documentation

6.177.5.1 Ipopt::TNLP::DECLARE_STD_EXCEPTION (INVALID_TNLP)

6.177.5.2 virtual bool Ipopt::TNLP::get_nlp_info (Index & n, Index & m, Index & nnz_jac_g, Index & nnz_h_lag, IndexStyleEnum & index_style) [pure virtual]

Implemented in [Ipopt::AmplTNLP](#), [Ipopt::StdInterfaceTNLP](#), and [Ipopt::TNLPReducer](#).

6.177.5.3 `virtual bool Ipopt::TNLP::get_var_con_metadata (Index n, StringMetaDataMapType & var_string_md, IntegerMetaDataMapType & var_integer_md, NumericMetaDataMapType & var_numeric_md, Index m, StringMetaDataMapType & con_string_md, IntegerMetaDataMapType & con_integer_md, NumericMetaDataMapType & con_numeric_md) [inline], [virtual]`

overload this method to return any meta data for the variables and the constraints

Reimplemented in [Ipopt::AmplTNLP](#).

Definition at line 90 of file IpTNLP.hpp.

6.177.5.4 `virtual bool Ipopt::TNLP::get_bounds_info (Index n, Number * x_l, Number * x_u, Index m, Number * g_l, Number * g_u) [pure virtual]`

overload this method to return the information about the bound on the variables and constraints.

The value that indicates that a bound does not exist is specified in the parameters `nlp_lower_bound_inf` and `nlp_upper_bound_inf`. By default, `nlp_lower_bound_inf` is -1e19 and `nlp_upper_bound_inf` is 1e19. (see [TNLPAdapter](#))

Implemented in [Ipopt::AmplTNLP](#), [Ipopt::StdInterfaceTNLP](#), and [Ipopt::TNLPReducer](#).

6.177.5.5 `virtual bool Ipopt::TNLP::get_scaling_parameters (Number & obj_scaling, bool & use_x_scaling, Index n, Number * x_scaling, bool & use_g_scaling, Index m, Number * g_scaling) [inline], [virtual]`

overload this method to return scaling parameters.

This is only called if the options are set to retrieve user scaling. There, `use_x_scaling` (or `use_g_scaling`) should get set to true only if the variables (or constraints) are to be scaled. This method should return true only if the scaling parameters could be provided.

Reimplemented in [Ipopt::AmplTNLP](#), [Ipopt::StdInterfaceTNLP](#), and [Ipopt::TNLPReducer](#).

Definition at line 119 of file IpTNLP.hpp.

6.177.5.6 `virtual bool Ipopt::TNLP::get_variables_linearity (Index n, LinearityType * var_types) [inline], [virtual]`

overload this method to return the variables linearity ([TNLP::LINEAR](#) or [TNLP::NON_LINEAR](#)).

The `var_types` array has been allocated with length at least n. (default implementation just return false and does not fill the array).

Reimplemented in [Ipopt::TNLPReducer](#).

Definition at line 132 of file IpTNLP.hpp.

6.177.5.7 `virtual bool Ipopt::TNLP::get_constraints_linearity (Index m, LinearityType * const_types) [inline], [virtual]`

overload this method to return the constraint linearity.

array has been allocated with length at least n. (default implementation just return false and does not fill the array).

Reimplemented in [Ipopt::AmplTNLP](#), and [Ipopt::TNLPReducer](#).

Definition at line 140 of file IpTNLP.hpp.

6.177.5.8 `virtual bool Ipopt::TNLP::get_starting_point (Index n, bool init_x, Number * x, bool init_z, Number * z_L, Number * z_U, Index m, bool init_lambda, Number * lambda) [pure virtual]`

overload this method to return the starting point.

The bool variables indicate whether the algorithm wants you to initialize x, z_L/z_u, and lambda, respectively. If, for

some reason, the algorithm wants you to initialize these and you cannot, return false, which will cause `lpopt` to stop. You will have to run `lpopt` with different options then.

Implemented in `lpopt::AmplTNLP`, `lpopt::StdInterfaceTNLP`, and `lpopt::TNLPReducer`.

6.177.5.9 `virtual bool lpopt::TNLP::get_warm_start_iterate (IteratesVector & warm_start_iterate) [inline], [virtual]`

overload this method to provide an `lpopt` iterate (already in the form `lpopt` requires it internally) for a warm start.

Since this is only for expert users, a default dummy implementation is provided and returns false.

Reimplemented in `lpopt::TNLPReducer`.

Definition at line 161 of file `lpTNLP.hpp`.

6.177.5.10 `virtual bool lpopt::TNLP::eval_f (Index n, const Number * x, bool new_x, Number & obj_value) [pure virtual]`

overload this method to return the value of the objective function

Implemented in `lpopt::AmplTNLP`, `lpopt::StdInterfaceTNLP`, and `lpopt::TNLPReducer`.

6.177.5.11 `virtual bool lpopt::TNLP::eval_grad_f (Index n, const Number * x, bool new_x, Number * grad_f) [pure virtual]`

overload this method to return the vector of the gradient of the objective w.r.t.

x

Implemented in `lpopt::AmplTNLP`, `lpopt::StdInterfaceTNLP`, and `lpopt::TNLPReducer`.

6.177.5.12 `virtual bool lpopt::TNLP::eval_g (Index n, const Number * x, bool new_x, Index m, Number * g) [pure virtual]`

overload this method to return the vector of constraint values

Implemented in `lpopt::AmplTNLP`, `lpopt::StdInterfaceTNLP`, and `lpopt::TNLPReducer`.

6.177.5.13 `virtual bool lpopt::TNLP::eval_jac_g (Index n, const Number * x, bool new_x, Index m, Index nele_jac, Index * iRow, Index * jCol, Number * values) [pure virtual]`

overload this method to return the jacobian of the constraints.

The vectors `iRow` and `jCol` only need to be set once. The first call is used to set the structure only (`iRow` and `jCol` will be non-NULL, and values will be NULL) For subsequent calls, `iRow` and `jCol` will be NULL.

Implemented in `lpopt::AmplTNLP`, `lpopt::StdInterfaceTNLP`, and `lpopt::TNLPReducer`.

6.177.5.14 `virtual bool lpopt::TNLP::eval_h (Index n, const Number * x, bool new_x, Number obj_factor, Index m, const Number * lambda, bool new_lambda, Index nele_hess, Index * iRow, Index * jCol, Number * values) [inline], [virtual]`

overload this method to return the hessian of the lagrangian.

The vectors `iRow` and `jCol` only need to be set once (during the first call). The first call is used to set the structure only (`iRow` and `jCol` will be non-NULL, and values will be NULL) For subsequent calls, `iRow` and `jCol` will be NULL. This matrix is symmetric - specify the lower diagonal only. A default implementation is provided, in case the user wants to se quasi-Newton approximations to estimate the second derivatives and doesn't not need to implement this method.

Reimplemented in `lpopt::AmplTNLP`, `lpopt::StdInterfaceTNLP`, and `lpopt::TNLPReducer`.

Definition at line 196 of file IpTNLP.hpp.

```
6.177.5.15 virtual void Ipopt::TNLP::finalize_solution ( SolverReturn status, Index n, const Number * x, const Number *
              z_L, const Number * z_U, Index m, const Number * g, const Number * lambda, Number obj_value, const
              IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq ) [pure virtual]
```

This method is called when the algorithm is complete so the [TNLP](#) can store/write the solution.

Implemented in [Ipopt::AmplTNLP](#), [Ipopt::StdInterfaceTNLP](#), and [Ipopt::TNLPReducer](#).

```
6.177.5.16 virtual void Ipopt::TNLP::finalize_metadata ( Index n, const StringMetaDataMapType & var_string_md, const
              IntegerMetaDataMapType & var_integer_md, const NumericMetaDataMapType & var_numeric_md, Index m,
              const StringMetaDataMapType & con_string_md, const IntegerMetaDataMapType & con_integer_md, const
              NumericMetaDataMapType & con_numeric_md ) [inline], [virtual]
```

This method is called just before `finalize_solution`.

With this method, the algorithm returns any metadata collected during its run, including the metadata provided by the user with the above `get_var_con_metadata`. Each metadata can be of type string, integer, and numeric. It can be associated to either the variables or the constraints. The metadata that was associated with the primal variable vector is stored in `var_..._md`. The metadata associated with the constraint multipliers is stored in `con_..._md`. The metadata associated with the bound multipliers is stored in `var_..._md`, with the suffixes "`_z_L`", and "`_z_U`", denoting lower and upper bounds.

Definition at line 226 of file IpTNLP.hpp.

```
6.177.5.17 virtual bool Ipopt::TNLP::intermediate_callback ( AlgorithmMode mode, Index iter, Number obj_value, Number
              inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du,
              Number alpha_pr, Index ls_trials, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq )
              [inline], [virtual]
```

Intermediate Callback method for the user.

Providing dummy default implementation. For details see `IntermediateCallback` in [IpNLP.hpp](#).

Reimplemented in [Ipopt::StdInterfaceTNLP](#), and [Ipopt::TNLPReducer](#).

Definition at line 240 of file IpTNLP.hpp.

```
6.177.5.18 virtual Index Ipopt::TNLP::get_number_of_nonlinear_variables ( ) [inline], [virtual]
```

Reimplemented in [Ipopt::AmplTNLP](#), and [Ipopt::TNLPReducer](#).

Definition at line 267 of file IpTNLP.hpp.

```
6.177.5.19 virtual bool Ipopt::TNLP::get_list_of_nonlinear_variables ( Index num_nonlin_vars, Index * pos_nonlin_vars )
              [inline], [virtual]
```

Reimplemented in [Ipopt::AmplTNLP](#), and [Ipopt::TNLPReducer](#).

Definition at line 272 of file IpTNLP.hpp.

```
6.177.5.20 void Ipopt::TNLP::operator= ( const TNLP & ) [private]
```

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

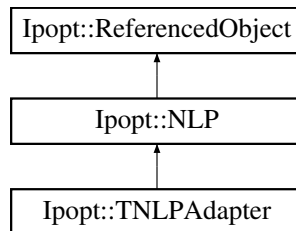
- [Interfaces/IpTNLP.hpp](#)

6.178 Ipopt::TNLPAdapter Class Reference

This class Adapts the [TNLP](#) interface so it looks like an [NLP](#) interface.

```
#include <IpTNLPAdapter.hpp>
```

Inheritance diagram for Ipopt::TNLPAdapter:



Public Types

- enum [FixedVariableTreatmentEnum](#) { [MAKE_PARAMETER](#) =0, [MAKE_CONSTRAINT](#), [RELAX_BOUNDS](#) }
Enum for treatment of fixed variables option.
- enum [DerivativeTestEnum](#) { [NO_TEST](#) =0, [FIRST_ORDER_TEST](#), [SECOND_ORDER_TEST](#), [ONLY_SECOND_ORDER_TEST](#) }
Enum for specifying which derivative test is to be performed.
- enum [JacobianApproxEnum](#) { [JAC_EXACT](#) =0, [JAC_FINDIFF_VALUES](#) }
Enum for specifying technique for computing Jacobian.

Public Member Functions

- virtual void [GetQuasiNewtonApproximationSpaces](#) ([SmartPtr](#)< [VectorSpace](#) > &approx_space, [SmartPtr](#)< [Matrix](#) > &P_approx)
Method returning information on quasi-Newton approximation.
- bool [CheckDerivatives](#) ([DerivativeTestEnum](#) deriv_test, [Index](#) deriv_test_start_index)
Method for performing the derivative test.
- [SmartPtr](#)< [TNLP](#) > [tnlp](#) () const
Accessor method for the underlying [TNLP](#).

Constructors/Destructors

- [TNLPAdapter](#) (const [SmartPtr](#)< [TNLP](#) > [tnlp](#), const [SmartPtr](#)< const [Journalist](#) > jnlst=NULL)
Default constructor.
- virtual [~TNLPAdapter](#) ()
Default destructor.

Exceptions

- [DECLARE_STD_EXCEPTION](#) (INVALID_TNLP)
- [DECLARE_STD_EXCEPTION](#) (ERROR_IN_TNLP_DERIVATIVE_TEST)

TNLPAdapter Initialization.

- virtual bool [ProcessOptions](#) (const [OptionsList](#) &options, const std::string &prefix)

Overload if you want the chance to process options or parameters that may be specific to the *NLP*.

- virtual bool `GetSpaces` (SmartPtr< const [VectorSpace](#) > &x_space, SmartPtr< const [VectorSpace](#) > &c_space, SmartPtr< const [VectorSpace](#) > &d_space, SmartPtr< const [VectorSpace](#) > &x_l_space, SmartPtr< const [MatrixSpace](#) > &px_l_space, SmartPtr< const [VectorSpace](#) > &x_u_space, SmartPtr< const [MatrixSpace](#) > &px_u_space, SmartPtr< const [VectorSpace](#) > &d_l_space, SmartPtr< const [MatrixSpace](#) > &pd_l_space, SmartPtr< const [VectorSpace](#) > &d_u_space, SmartPtr< const [MatrixSpace](#) > &pd_u_space, SmartPtr< const [MatrixSpace](#) > &Jac_c_space, SmartPtr< const [MatrixSpace](#) > &Jac_d_space, SmartPtr< const [SymMatrixSpace](#) > &Hess_lagrangian_space)

Method for creating the derived vector / matrix types (Do not delete these, the).

- virtual bool `GetBoundsInformation` (const [Matrix](#) &Px_L, [Vector](#) &x_L, const [Matrix](#) &Px_U, [Vector](#) &x_U, const [Matrix](#) &Pd_L, [Vector](#) &d_L, const [Matrix](#) &Pd_U, [Vector](#) &d_U)

Method for obtaining the bounds information.

- virtual bool `GetStartingPoint` (SmartPtr< [Vector](#) > x, bool need_x, SmartPtr< [Vector](#) > y_c, bool need_y_c, SmartPtr< [Vector](#) > y_d, bool need_y_d, SmartPtr< [Vector](#) > z_L, bool need_z_L, SmartPtr< [Vector](#) > z_U, bool need_z_U)

Method for obtaining the starting point for all the iterates.

- virtual bool `GetWarmStartIterate` ([IteratesVector](#) &warm_start_iterate)

Method for obtaining an entire iterate as a warmstart point.

TNLPAdapter evaluation routines.

- virtual bool `Eval_f` (const [Vector](#) &x, [Number](#) &f)
- virtual bool `Eval_grad_f` (const [Vector](#) &x, [Vector](#) &g_f)
- virtual bool `Eval_c` (const [Vector](#) &x, [Vector](#) &c)
- virtual bool `Eval_jac_c` (const [Vector](#) &x, [Matrix](#) &jac_c)
- virtual bool `Eval_d` (const [Vector](#) &x, [Vector](#) &d)
- virtual bool `Eval_jac_d` (const [Vector](#) &x, [Matrix](#) &jac_d)
- virtual bool `Eval_h` (const [Vector](#) &x, [Number](#) obj_factor, const [Vector](#) &yc, const [Vector](#) &yd, [SymMatrix](#) &h)
- virtual void `GetScalingParameters` (const SmartPtr< const [VectorSpace](#) > x_space, const SmartPtr< const [VectorSpace](#) > c_space, const SmartPtr< const [VectorSpace](#) > d_space, [Number](#) &obj_scaling, SmartPtr< [Vector](#) > &x_scaling, SmartPtr< [Vector](#) > &c_scaling, SmartPtr< [Vector](#) > &d_scaling) const

Routines to get the scaling parameters.

Solution Reporting Methods

- virtual void `FinalizeSolution` ([SolverReturn](#) status, const [Vector](#) &x, const [Vector](#) &z_L, const [Vector](#) &z_U, const [Vector](#) &c, const [Vector](#) &d, const [Vector](#) &y_c, const [Vector](#) &y_d, [Number](#) obj_value, const [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)

This method is called at the very end of the optimization.

- virtual bool `IntermediateCallBack` ([AlgorithmMode](#) mode, [Index](#) iter, [Number](#) obj_value, [Number](#) inf_pr, [Number](#) inf_du, [Number](#) mu, [Number](#) d_norm, [Number](#) regularization_size, [Number](#) alpha_du, [Number](#) alpha_pr, [Index](#) ls_trials, const [IpoptData](#) *ip_data, [IpoptCalculatedQuantities](#) *ip_cq)

This method is called once per iteration, after the iteration summary output has been printed.

Methods for translating data for IpoptNLP into the TNLP

data.

These methods are used to obtain the current (or final) data for the *TNLP* formulation from the *IpoptNLP* structure.

- void `ResortX` (const [Vector](#) &x, [Number](#) *x_orig)
Sort the primal variables, and add the fixed values in x.
- void `ResortG` (const [Vector](#) &c, const [Vector](#) &d, [Number](#) *g_orig)
- void `ResortBnds` (const [Vector](#) &x_L, [Number](#) *x_L_orig, const [Vector](#) &x_U, [Number](#) *x_U_orig)

Static Public Member Functions

Methods for IpoptType

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TNLPAdapter](#) (const [TNLPAdapter](#) &)
Copy Constructor.
- void [operator=](#) (const [TNLPAdapter](#) &)
Overloaded Equals Operator.

Methods to update the values in the local copies of vectors

- bool [update_local_x](#) (const [Vector](#) &x)
- bool [update_local_lambda](#) (const [Vector](#) &y_c, const [Vector](#) &y_d)

Internal routines for evaluating g and jac_g (values stored since

they are used in both c and d routines

- bool [internal_eval_g](#) (bool new_x)
- bool [internal_eval_jac_g](#) (bool new_x)

Internal methods for dealing with finite difference

approximation

- void [initialize_findiff_jac](#) (const [Index](#) *iRow, const [Index](#) *jCol)
Initialize sparsity structure for finite difference Jacobian.

Private Attributes

- [TNLP::IndexStyleEnum index_style_](#)
Numbering style of variables and constraints.

Algorithmic parameters

- [Number nlp_lower_bound_inf_](#)
Value for a lower bound that denotes -infinity.
- [Number nlp_upper_bound_inf_](#)
Value for a upper bound that denotes infinity.
- [FixedVariableTreatmentEnum fixed_variable_treatment_](#)
Flag indicating how fixed variables should be handled.
- [Number bound_relax_factor_](#)
- [DerivativeTestEnum derivative_test_](#)
Enum indicating whether and which derivative test should be performed at starting point.
- [Number derivative_test_perturbation_](#)

- *Size of the perturbation for the derivative test.*
- [Number derivative_test_tol_](#)
Relative threshold for marking deviation from finite difference test.
- [bool derivative_test_print_all_](#)
Flag indicating if all test values should be printed, or only those violating the threshold.
- [Index derivative_test_first_index_](#)
Index of first quantity to be checked.
- [bool warm_start_same_structure_](#)
*Flag indicating whether the *TNLP* with identical structure has already been solved before.*
- [HessianApproximationType hessian_approximation_](#)
Flag indicating what Hessian information is to be used.
- [Index num_linear_variables_](#)
Number of linear variables.
- [JacobianApproxEnum jacobian_approximation_](#)
Flag indicating how Jacobian is computed.
- [Number findiff_perturbation_](#)
Size of the perturbation for the derivative approximation.
- [Number point_perturbation_radius_](#)
Maximal perturbation of the initial point.
- [bool dependency_detection_with_rhs_](#)
Flag indicating if rhs should be considered during dependency detection.
- [Number tol_](#)
Overall convergence tolerance.

Problem Size Data

- [Index n_full_x_](#)
full dimension of x (fixed + non-fixed)
- [Index n_full_g_](#)
full dimension of g ($c + d$)
- [Index nz_jac_c_](#)
non-zeros of the jacobian of c
- [Index nz_jac_c_no_extra_](#)
non-zeros of the jacobian of c without added constraints for fixed variables.
- [Index nz_jac_d_](#)
non-zeros of the jacobian of d
- [Index nz_full_jac_g_](#)
number of non-zeros in full-size Jacobian of g
- [Index nz_full_h_](#)
number of non-zeros in full-size Hessian
- [Index nz_h_](#)
number of non-zeros in the non-fixed-size Hessian
- [Index n_x_fixed_](#)
Number of fixed variables.

Local copy of spaces (for warm start)

- [SmartPtr< const VectorSpace > x_space_](#)
- [SmartPtr< const VectorSpace > c_space_](#)
- [SmartPtr< const VectorSpace > d_space_](#)
- [SmartPtr< const VectorSpace > x_l_space_](#)
- [SmartPtr< const MatrixSpace > px_l_space_](#)
- [SmartPtr< const VectorSpace > x_u_space_](#)
- [SmartPtr< const MatrixSpace > px_u_space_](#)
- [SmartPtr< const VectorSpace > d_l_space_](#)

- [SmartPtr< const MatrixSpace > pd_l_space_](#)
- [SmartPtr< const VectorSpace > d_u_space_](#)
- [SmartPtr< const MatrixSpace > pd_u_space_](#)
- [SmartPtr< const MatrixSpace > Jac_c_space_](#)
- [SmartPtr< const MatrixSpace > Jac_d_space_](#)
- [SmartPtr< const SymMatrixSpace > Hess_lagrangian_space_](#)

Local Copy of the Data

- [Number * full_x_](#)
- [Number * full_lambda_](#)
copy of the full x vector (fixed & non-fixed)
- [Number * full_g_](#)
copy of lambda (yc & yd)
- [Number * jac_g_](#)
copy of g (c & d)
- [Number * c_rhs_](#)
the values for the full jacobian of g

Tags for deciding when to update internal copies of vectors

the rhs values of c

- [TaggedObject::Tag x_tag_for_iterates_](#)
- [TaggedObject::Tag y_c_tag_for_iterates_](#)
- [TaggedObject::Tag y_d_tag_for_iterates_](#)
- [TaggedObject::Tag x_tag_for_g_](#)
- [TaggedObject::Tag x_tag_for_jac_g_](#)

Internal Permutation Spaces and matrices

- [SmartPtr< ExpansionMatrix > P_x_full_x_](#)
Expansion from fixed x (ipopt) to full x.
- [SmartPtr< ExpansionMatrixSpace > P_x_full_x_space_](#)
- [SmartPtr< ExpansionMatrix > P_x_x_L_](#)
Expansion from fixed x_L (ipopt) to full x.
- [SmartPtr< ExpansionMatrixSpace > P_x_x_L_space_](#)
- [SmartPtr< ExpansionMatrix > P_x_x_U_](#)
Expansion from fixed x_U (ipopt) to full x.
- [SmartPtr< ExpansionMatrixSpace > P_x_x_U_space_](#)
- [SmartPtr< ExpansionMatrixSpace > P_c_g_space_](#)
Expansion from c only (ipopt) to full ampl c.
- [SmartPtr< ExpansionMatrix > P_c_g_](#)
- [SmartPtr< ExpansionMatrixSpace > P_d_g_space_](#)
Expansion from d only (ipopt) to full ampl d.
- [SmartPtr< ExpansionMatrix > P_d_g_](#)
- [Index * jac_idx_map_](#)
- [Index * h_idx_map_](#)
- [Index * x_fixed_map_](#)
Position of fixed variables.

Data for finite difference approximations of derivatives

- [Index findiff_jac_nnz_](#)
Number of unique nonzeros in constraint Jacobian.
- [Index * findiff_jac_ja_](#)

- Start position for nonzero indices in `ja` for each column of *Jacobian*
- [Index](#) * `findiff_jac_ja_`
Ordered by columns, for each column the row indices in *Jacobian*
- [Index](#) * `findiff_jac_postriplet_`
Position of entry in original triplet matrix.
- [Number](#) * `findiff_x_l_`
Copy of the lower bounds.
- [Number](#) * `findiff_x_u_`
Copy of the upper bounds.

Method implementing the detection of linearly dependent

equality constraints

- [SmartPtr](#)< [TNLP](#) > `tnlp_`
Pointer to the [TNLP](#) class (class specific to [Number](#)* vectors and harwell triplet matrices)
- [SmartPtr](#)< const [Journalist](#) > `jnlst_`
Journalist.
- [SmartPtr](#)< [TDependencyDetector](#) > `dependency_detector_`
Object that can be used to detect linearly dependent rows in the equality constraint *Jacobian*.
- bool `DetermineDependentConstraints` ([Index](#) `n_x_var`, const [Index](#) *`x_not_fixed_map`, const [Number](#) *`x_l`, const [Number](#) *`x_u`, const [Number](#) *`g_l`, const [Number](#) *`g_u`, [Index](#) `n_c`, const [Index](#) *`c_map`, std::list< [Index](#) > &`c_deps`)

6.178.1 Detailed Description

This class Adapts the [TNLP](#) interface so it looks like an [NLP](#) interface.

This is an Adapter class (Design Patterns) that converts a [TNLP](#) to an [NLP](#). This allows users to write to the "more convenient" [TNLP](#) interface.

Definition at line 30 of file `IpTNLPAdapter.hpp`.

6.178.2 Member Enumeration Documentation

6.178.2.1 enum Ipopt::TNLPAdapter::FixedVariableTreatmentEnum

Enum for treatment of fixed variables option.

Enumerator

`MAKE_PARAMETER`
`MAKE_CONSTRAINT`
`RELAX_BOUNDS`

Definition at line 159 of file `IpTNLPAdapter.hpp`.

6.178.2.2 enum `Ipopt::TNLPAdapter::DerivativeTestEnum`

Enum for specifying which derivative test is to be performed.

Enumerator

`NO_TEST`
`FIRST_ORDER_TEST`
`SECOND_ORDER_TEST`
`ONLY_SECOND_ORDER_TEST`

Definition at line 167 of file `IpTNLPAdapter.hpp`.

6.178.2.3 enum `Ipopt::TNLPAdapter::JacobianApproxEnum`

Enum for specifying technique for computing Jacobian.

Enumerator

`JAC_EXACT`
`JAC_FINDIFF_VALUES`

Definition at line 176 of file `IpTNLPAdapter.hpp`.

6.178.3 Constructor & Destructor Documentation

6.178.3.1 `Ipopt::TNLPAdapter::TNLPAdapter (const SmartPtr< TNLP > tnlp, const SmartPtr< const Journalist > jnlst = NULL)`

Default constructor.

6.178.3.2 `virtual Ipopt::TNLPAdapter::~TNLPAdapter () [virtual]`

Default destructor.

6.178.3.3 `Ipopt::TNLPAdapter::TNLPAdapter (const TNLPAdapter &) [private]`

Copy Constructor.

6.178.4 Member Function Documentation

6.178.4.1 `Ipopt::TNLPAdapter::DECLARE_STD_EXCEPTION (INVALID_TNLP)`

6.178.4.2 `Ipopt::TNLPAdapter::DECLARE_STD_EXCEPTION (ERROR_IN_TNLP_DERIVATIVE_TEST)`

6.178.4.3 `virtual bool Ipopt::TNLPAdapter::ProcessOptions (const OptionsList & options, const std::string & prefix) [virtual]`

Overload if you want the chance to process options or parameters that may be specific to the [NLP](#).

Reimplemented from [Ipopt::NLP](#).

6.178.4.4 `virtual bool Ipopt::TNLPAdapter::GetSpaces (SmartPtr< const VectorSpace > & x_space, SmartPtr< const VectorSpace > & c_space, SmartPtr< const VectorSpace > & d_space, SmartPtr< const VectorSpace > & x_L_space, SmartPtr< const MatrixSpace > & px_L_space, SmartPtr< const VectorSpace > & x_U_space, SmartPtr< const MatrixSpace > & px_U_space, SmartPtr< const VectorSpace > & d_L_space, SmartPtr< const MatrixSpace > & pd_L_space, SmartPtr< const VectorSpace > & d_U_space, SmartPtr< const MatrixSpace > & pd_U_space, SmartPtr< const MatrixSpace > & Jac_c_space, SmartPtr< const MatrixSpace > & Jac_d_space, SmartPtr< const SymMatrixSpace > & Hess_lagrangian_space) [virtual]`

Method for creating the derived vector / matrix types (Do not delete these, the).

Implements [Ipopt::NLP](#).

6.178.4.5 `virtual bool Ipopt::TNLPAdapter::GetBoundsInformation (const Matrix & Px_L, Vector & x_L, const Matrix & Px_U, Vector & x_U, const Matrix & Pd_L, Vector & d_L, const Matrix & Pd_U, Vector & d_U) [virtual]`

Method for obtaining the bounds information.

Implements [Ipopt::NLP](#).

6.178.4.6 `virtual bool Ipopt::TNLPAdapter::GetStartingPoint (SmartPtr< Vector > x, bool need_x, SmartPtr< Vector > y_c, bool need_y_c, SmartPtr< Vector > y_d, bool need_y_d, SmartPtr< Vector > z_L, bool need_z_L, SmartPtr< Vector > z_U, bool need_z_U) [virtual]`

Method for obtaining the starting point for all the iterates.

Implements [Ipopt::NLP](#).

6.178.4.7 `virtual bool Ipopt::TNLPAdapter::GetWarmStartIterate (IteratesVector & warm_start_iterate) [virtual]`

Method for obtaining an entire iterate as a warmstart point.

The incoming [IteratesVector](#) has to be filled.

Reimplemented from [Ipopt::NLP](#).

6.178.4.8 `virtual bool Ipopt::TNLPAdapter::Eval_f (const Vector & x, Number & f) [virtual]`

Implements [Ipopt::NLP](#).

6.178.4.9 `virtual bool Ipopt::TNLPAdapter::Eval_grad_f (const Vector & x, Vector & g_f) [virtual]`

Implements [Ipopt::NLP](#).

6.178.4.10 `virtual bool Ipopt::TNLPAdapter::Eval_c (const Vector & x, Vector & c) [virtual]`

Implements [Ipopt::NLP](#).

6.178.4.11 `virtual bool Ipopt::TNLPAdapter::Eval_jac_c (const Vector & x, Matrix & jac_c) [virtual]`

Implements [Ipopt::NLP](#).

6.178.4.12 `virtual bool Ipopt::TNLPAdapter::Eval_d (const Vector & x, Vector & d) [virtual]`

Implements [Ipopt::NLP](#).

6.178.4.13 `virtual bool Ipopt::TNLPAdapter::Eval_jac_d (const Vector & x, Matrix & jac_d) [virtual]`

Implements [Ipopt::NLP](#).

6.178.4.14 `virtual bool Ipopt::TNLPAdapter::Eval_h (const Vector & x, Number obj_factor, const Vector & yc, const Vector & yd, SymMatrix & h) [virtual]`

Implements [Ipopt::NLP](#).

6.178.4.15 `virtual void Ipopt::TNLPAdapter::GetScalingParameters (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, Number & obj_scaling, SmartPtr< Vector > & x_scaling, SmartPtr< Vector > & c_scaling, SmartPtr< Vector > & d_scaling) const [virtual]`

Routines to get the scaling parameters.

These do not need to be overloaded unless the options are set for User scaling

Reimplemented from [Ipopt::NLP](#).

6.178.4.16 `virtual void Ipopt::TNLPAdapter::FinalizeSolution (SolverReturn status, const Vector & x, const Vector & z_L, const Vector & z_U, const Vector & c, const Vector & d, const Vector & y_c, const Vector & y_d, Number obj_value, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [virtual]`

This method is called at the very end of the optimization.

It provides the final iterate to the user, so that it can be stored as the solution. The status flag indicates the outcome of the optimization, where SolverReturn is defined in [IpAlgTypes.hpp](#).

Reimplemented from [Ipopt::NLP](#).

6.178.4.17 `virtual bool Ipopt::TNLPAdapter::IntermediateCallBack (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq) [virtual]`

This method is called once per iteration, after the iteration summary output has been printed.

It provides the current information to the user to do with it anything she wants. It also allows the user to ask for a premature termination of the optimization by returning false, in which case [Ipopt](#) will terminate with a corresponding return status. The basic information provided in the argument list has the quantities values printed in the iteration summary line. If more information is required, a user can obtain it from the IpData and IpCalculatedQuantities objects. However, note that the provided quantities are all for the problem that [Ipopt](#) sees, i.e., the quantities might be scaled, fixed variables might be sorted out, etc. The status indicates things like whether the algorithm is in the restoration phase... In the restoration phase, the dual variables are probably not changing.

Reimplemented from [Ipopt::NLP](#).

6.178.4.18 `virtual void Ipopt::TNLPAdapter::GetQuasiNewtonApproximationSpaces (SmartPtr< VectorSpace > & approx_space, SmartPtr< Matrix > & P_approx) [virtual]`

Method returning information on quasi-Newton approximation.

Reimplemented from [Ipopt::NLP](#).

6.178.4.19 `bool Ipopt::TNLPAdapter::CheckDerivatives (DerivativeTestEnum deriv_test, Index deriv_test_start_index)`

Method for performing the derivative test.

6.178.4.20 `static void Ipopt::TNLPAdapter::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

6.178.4.21 `SmartPtr<TNLP> Ipopt::TNLPAdapter::tnlp () const [inline]`

Accessor method for the underlying [TNLP](#).

Definition at line 192 of file IpTNLPAdapter.hpp.

6.178.4.22 void Ipopt::TNLPAdapter::ResortX (const Vector & x, Number * x_orig)

Sort the primal variables, and add the fixed values in x.

6.178.4.23 void Ipopt::TNLPAdapter::ResortG (const Vector & c, const Vector & d, Number * g_orig)

6.178.4.24 void Ipopt::TNLPAdapter::ResortBnds (const Vector & x_L, Number * x_L_orig, const Vector & x_U, Number * x_U_orig)

6.178.4.25 void Ipopt::TNLPAdapter::operator= (const TNLPAdapter &) [private]

Overloaded Equals Operator.

6.178.4.26 bool Ipopt::TNLPAdapter::DetermineDependentConstraints (Index n_x_var, const Index * x_not_fixed_map, const Number * x_l, const Number * x_u, const Number * g_l, const Number * g_u, Index n_c, const Index * c_map, std::list< Index > & c_deps) [private]

6.178.4.27 bool Ipopt::TNLPAdapter::update_local_x (const Vector & x) [private]

6.178.4.28 bool Ipopt::TNLPAdapter::update_local_lambda (const Vector & y_c, const Vector & y_d) [private]

6.178.4.29 bool Ipopt::TNLPAdapter::internal_eval_g (bool new_x) [private]

6.178.4.30 bool Ipopt::TNLPAdapter::internal_eval_jac_g (bool new_x) [private]

6.178.4.31 void Ipopt::TNLPAdapter::initialize_findiff_jac (const Index * iRow, const Index * jCol) [private]

Initialize sparsity structure for finite difference Jacobian.

6.178.5 Member Data Documentation

6.178.5.1 SmartPtr<TNLP> Ipopt::TNLPAdapter::tnlp_ [private]

Pointer to the [TNLP](#) class (class specific to Number* vectors and harwell triplet matrices)

Definition at line 236 of file IpTNLPAdapter.hpp.

6.178.5.2 SmartPtr<const Journalist> Ipopt::TNLPAdapter::jnlst_ [private]

[Journalist](#).

Definition at line 239 of file IpTNLPAdapter.hpp.

6.178.5.3 SmartPtr<TDependencyDetector> Ipopt::TNLPAdapter::dependency_detector_ [private]

Object that can be used to detect linearly dependent rows in the equality constraint Jacobian.

Definition at line 243 of file IpTNLPAdapter.hpp.

6.178.5.4 Number Ipopt::TNLPAdapter::nlp_lower_bound_inf_ [private]

Value for a lower bound that denotes -infinity.

Definition at line 248 of file IpTNLPAdapter.hpp.

6.178.5.5 Number `Ipopt::TNLPAdapter::nlp_upper_bound_inf_` `[private]`

Value for a upper bound that denotes infinity.

Definition at line 250 of file `IpTNLPAdapter.hpp`.

6.178.5.6 FixedVariableTreatmentEnum `Ipopt::TNLPAdapter::fixed_variable_treatment_` `[private]`

Flag indicating how fixed variables should be handled.

Definition at line 252 of file `IpTNLPAdapter.hpp`.

6.178.5.7 Number `Ipopt::TNLPAdapter::bound_relax_factor_` `[private]`

Definition at line 254 of file `IpTNLPAdapter.hpp`.

6.178.5.8 DerivativeTestEnum `Ipopt::TNLPAdapter::derivative_test_` `[private]`

Enum indicating whether and which derivative test should be performed at starting point.

Definition at line 262 of file `IpTNLPAdapter.hpp`.

6.178.5.9 Number `Ipopt::TNLPAdapter::derivative_test_perturbation_` `[private]`

Size of the perturbation for the derivative test.

Definition at line 264 of file `IpTNLPAdapter.hpp`.

6.178.5.10 Number `Ipopt::TNLPAdapter::derivative_test_tol_` `[private]`

Relative threshold for marking deviation from finite difference test.

Definition at line 267 of file `IpTNLPAdapter.hpp`.

6.178.5.11 bool `Ipopt::TNLPAdapter::derivative_test_print_all_` `[private]`

Flag indicating if all test values should be printed, or only those violating the threshold.

Definition at line 270 of file `IpTNLPAdapter.hpp`.

6.178.5.12 Index `Ipopt::TNLPAdapter::derivative_test_first_index_` `[private]`

Index of first quantity to be checked.

Definition at line 272 of file `IpTNLPAdapter.hpp`.

6.178.5.13 bool `Ipopt::TNLPAdapter::warm_start_same_structure_` `[private]`

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 275 of file `IpTNLPAdapter.hpp`.

6.178.5.14 HessianApproximationType `Ipopt::TNLPAdapter::hessian_approximation_` `[private]`

Flag indicating what Hessian information is to be used.

Definition at line 277 of file `IpTNLPAdapter.hpp`.

6.178.5.15 Index `Ipopt::TNLPAdapter::num_linear_variables_` `[private]`

Number of linear variables.

Definition at line 279 of file IpTNLPAdapter.hpp.

6.178.5.16 JacobianApproxEnum Ipopt::TNLPAdapter::jacobian_approximation_ [private]

Flag indicating how Jacobian is computed.

Definition at line 281 of file IpTNLPAdapter.hpp.

6.178.5.17 Number Ipopt::TNLPAdapter::findiff_perturbation_ [private]

Size of the perturbation for the derivative approximation.

Definition at line 283 of file IpTNLPAdapter.hpp.

6.178.5.18 Number Ipopt::TNLPAdapter::point_perturbation_radius_ [private]

Maximal perturbation of the initial point.

Definition at line 285 of file IpTNLPAdapter.hpp.

6.178.5.19 bool Ipopt::TNLPAdapter::dependency_detection_with_rhs_ [private]

Flag indicating if rhs should be considered during dependency detection.

Definition at line 288 of file IpTNLPAdapter.hpp.

6.178.5.20 Number Ipopt::TNLPAdapter::tol_ [private]

Overall convergence tolerance.

Definition at line 291 of file IpTNLPAdapter.hpp.

6.178.5.21 Index Ipopt::TNLPAdapter::n_full_x_ [private]

full dimension of x (fixed + non-fixed)

Definition at line 297 of file IpTNLPAdapter.hpp.

6.178.5.22 Index Ipopt::TNLPAdapter::n_full_g_ [private]

full dimension of g (c + d)

Definition at line 299 of file IpTNLPAdapter.hpp.

6.178.5.23 Index Ipopt::TNLPAdapter::nz_jac_c_ [private]

non-zeros of the jacobian of c

Definition at line 301 of file IpTNLPAdapter.hpp.

6.178.5.24 Index Ipopt::TNLPAdapter::nz_jac_c_no_extra_ [private]

non-zeros of the jacobian of c without added constraints for fixed variables.

Definition at line 304 of file IpTNLPAdapter.hpp.

6.178.5.25 Index Ipopt::TNLPAdapter::nz_jac_d_ [private]

non-zeros of the jacobian of d

Definition at line 306 of file IpTNLPAdapter.hpp.

6.178.5.26 Index Ipopt::TNLPAdapter::nz_full_jac_g_ [private]

number of non-zeros in full-size Jacobian of g

Definition at line 308 of file IpTNLPAdapter.hpp.

6.178.5.27 Index Ipopt::TNLPAdapter::nz_full_h_ [private]

number of non-zeros in full-size Hessian

Definition at line 310 of file IpTNLPAdapter.hpp.

6.178.5.28 Index Ipopt::TNLPAdapter::nz_h_ [private]

number of non-zeros in the non-fixed-size Hessian

Definition at line 312 of file IpTNLPAdapter.hpp.

6.178.5.29 Index Ipopt::TNLPAdapter::n_x_fixed_ [private]

Number of fixed variables.

Definition at line 314 of file IpTNLPAdapter.hpp.

6.178.5.30 TNLP::IndexStyleEnum Ipopt::TNLPAdapter::index_style_ [private]

Numbering style of variables and constraints.

Definition at line 318 of file IpTNLPAdapter.hpp.

6.178.5.31 SmartPtr<const VectorSpace> Ipopt::TNLPAdapter::x_space_ [private]

Definition at line 322 of file IpTNLPAdapter.hpp.

6.178.5.32 SmartPtr<const VectorSpace> Ipopt::TNLPAdapter::c_space_ [private]

Definition at line 323 of file IpTNLPAdapter.hpp.

6.178.5.33 SmartPtr<const VectorSpace> Ipopt::TNLPAdapter::d_space_ [private]

Definition at line 324 of file IpTNLPAdapter.hpp.

6.178.5.34 SmartPtr<const VectorSpace> Ipopt::TNLPAdapter::x_l_space_ [private]

Definition at line 325 of file IpTNLPAdapter.hpp.

6.178.5.35 SmartPtr<const MatrixSpace> Ipopt::TNLPAdapter::px_l_space_ [private]

Definition at line 326 of file IpTNLPAdapter.hpp.

6.178.5.36 SmartPtr<const VectorSpace> Ipopt::TNLPAdapter::x_u_space_ [private]

Definition at line 327 of file IpTNLPAdapter.hpp.

6.178.5.37 SmartPtr<const MatrixSpace> Ipopt::TNLPAdapter::px_u_space_ [private]

Definition at line 328 of file IpTNLPAdapter.hpp.

6.178.5.38 `SmartPtr<const VectorSpace> Ipopt::TNLPAdapter::d_l_space_` [private]

Definition at line 329 of file IpTNLPAdapter.hpp.

6.178.5.39 `SmartPtr<const MatrixSpace> Ipopt::TNLPAdapter::pd_l_space_` [private]

Definition at line 330 of file IpTNLPAdapter.hpp.

6.178.5.40 `SmartPtr<const VectorSpace> Ipopt::TNLPAdapter::d_u_space_` [private]

Definition at line 331 of file IpTNLPAdapter.hpp.

6.178.5.41 `SmartPtr<const MatrixSpace> Ipopt::TNLPAdapter::pd_u_space_` [private]

Definition at line 332 of file IpTNLPAdapter.hpp.

6.178.5.42 `SmartPtr<const MatrixSpace> Ipopt::TNLPAdapter::Jac_c_space_` [private]

Definition at line 333 of file IpTNLPAdapter.hpp.

6.178.5.43 `SmartPtr<const MatrixSpace> Ipopt::TNLPAdapter::Jac_d_space_` [private]

Definition at line 334 of file IpTNLPAdapter.hpp.

6.178.5.44 `SmartPtr<const SymMatrixSpace> Ipopt::TNLPAdapter::Hess_lagrangian_space_` [private]

Definition at line 335 of file IpTNLPAdapter.hpp.

6.178.5.45 `Number* Ipopt::TNLPAdapter::full_x_` [private]

Definition at line 340 of file IpTNLPAdapter.hpp.

6.178.5.46 `Number* Ipopt::TNLPAdapter::full_lambda_` [private]

copy of the full x vector (fixed & non-fixed)

Definition at line 341 of file IpTNLPAdapter.hpp.

6.178.5.47 `Number* Ipopt::TNLPAdapter::full_g_` [private]

copy of lambda (yc & yd)

Definition at line 342 of file IpTNLPAdapter.hpp.

6.178.5.48 `Number* Ipopt::TNLPAdapter::jac_g_` [private]

copy of g (c & d)

Definition at line 343 of file IpTNLPAdapter.hpp.

6.178.5.49 `Number* Ipopt::TNLPAdapter::c_rhs_` [private]

the values for the full jacobian of g

Definition at line 344 of file IpTNLPAdapter.hpp.

6.178.5.50 `TaggedObject::Tag Ipopt::TNLPAdapter::x_tag_for_iterates_` [private]

Definition at line 349 of file IpTNLPAdapter.hpp.

6.178.5.51 TaggedObject::Tag Ipopt::TNLPAdapter::y_c_tag_for_iterates_ [private]

Definition at line 350 of file IpTNLPAdapter.hpp.

6.178.5.52 TaggedObject::Tag Ipopt::TNLPAdapter::y_d_tag_for_iterates_ [private]

Definition at line 351 of file IpTNLPAdapter.hpp.

6.178.5.53 TaggedObject::Tag Ipopt::TNLPAdapter::x_tag_for_g_ [private]

Definition at line 352 of file IpTNLPAdapter.hpp.

6.178.5.54 TaggedObject::Tag Ipopt::TNLPAdapter::x_tag_for_jac_g_ [private]

Definition at line 353 of file IpTNLPAdapter.hpp.

6.178.5.55 SmartPtr<ExpansionMatrix> Ipopt::TNLPAdapter::P_x_full_x_ [private]

Expansion from fixed x (ipopt) to full x.

Definition at line 380 of file IpTNLPAdapter.hpp.

6.178.5.56 SmartPtr<ExpansionMatrixSpace> Ipopt::TNLPAdapter::P_x_full_x_space_ [private]

Definition at line 381 of file IpTNLPAdapter.hpp.

6.178.5.57 SmartPtr<ExpansionMatrix> Ipopt::TNLPAdapter::P_x_x_L_ [private]

Expansion from fixed x_L (ipopt) to full x.

Definition at line 384 of file IpTNLPAdapter.hpp.

6.178.5.58 SmartPtr<ExpansionMatrixSpace> Ipopt::TNLPAdapter::P_x_x_L_space_ [private]

Definition at line 385 of file IpTNLPAdapter.hpp.

6.178.5.59 SmartPtr<ExpansionMatrix> Ipopt::TNLPAdapter::P_x_x_U_ [private]

Expansion from fixed x_U (ipopt) to full x.

Definition at line 388 of file IpTNLPAdapter.hpp.

6.178.5.60 SmartPtr<ExpansionMatrixSpace> Ipopt::TNLPAdapter::P_x_x_U_space_ [private]

Definition at line 389 of file IpTNLPAdapter.hpp.

6.178.5.61 SmartPtr<ExpansionMatrixSpace> Ipopt::TNLPAdapter::P_c_g_space_ [private]

Expansion from c only (ipopt) to full ampl c.

Definition at line 392 of file IpTNLPAdapter.hpp.

6.178.5.62 SmartPtr<ExpansionMatrix> Ipopt::TNLPAdapter::P_c_g_ [private]

Definition at line 393 of file IpTNLPAdapter.hpp.

6.178.5.63 SmartPtr<ExpansionMatrixSpace> Ipopt::TNLPAdapter::P_d_g_space_ [private]

Expansion from d only (ipopt) to full ampl d.

Definition at line 396 of file IpTNLPAdapter.hpp.

6.178.5.64 `SmartPtr<ExpansionMatrix> Ipopt::TNLPAdapter::P_d_g_ [private]`

Definition at line 397 of file IpTNLPAdapter.hpp.

6.178.5.65 `Index* Ipopt::TNLPAdapter::jac_idx_map_ [private]`

Definition at line 399 of file IpTNLPAdapter.hpp.

6.178.5.66 `Index* Ipopt::TNLPAdapter::h_idx_map_ [private]`

Definition at line 400 of file IpTNLPAdapter.hpp.

6.178.5.67 `Index* Ipopt::TNLPAdapter::x_fixed_map_ [private]`

Position of fixed variables.

This is required for a warm start

Definition at line 403 of file IpTNLPAdapter.hpp.

6.178.5.68 `Index Ipopt::TNLPAdapter::findiff_jac_nnz_ [private]`

Number of unique nonzeros in constraint Jacobian.

Definition at line 409 of file IpTNLPAdapter.hpp.

6.178.5.69 `Index* Ipopt::TNLPAdapter::findiff_jac_ia_ [private]`

Start position for nonzero indices in ja for each column of

Jacobian

Definition at line 412 of file IpTNLPAdapter.hpp.

6.178.5.70 `Index* Ipopt::TNLPAdapter::findiff_jac_ja_ [private]`

Ordered by columns, for each column the row indices in

Jacobian

Definition at line 415 of file IpTNLPAdapter.hpp.

6.178.5.71 `Index* Ipopt::TNLPAdapter::findiff_jac_postriplet_ [private]`

Position of entry in original triplet matrix.

Definition at line 417 of file IpTNLPAdapter.hpp.

6.178.5.72 `Number* Ipopt::TNLPAdapter::findiff_x_l_ [private]`

Copy of the lower bounds.

Definition at line 419 of file IpTNLPAdapter.hpp.

6.178.5.73 `Number* Ipopt::TNLPAdapter::findiff_x_u_ [private]`

Copy of the upper bounds.

Definition at line 421 of file IpTNLPAdapter.hpp.

The documentation for this class was generated from the following file:

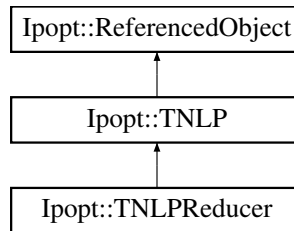
- Interfaces/[IpTNLPAdapter.hpp](#)

6.179 Ipopt::TNLPReducer Class Reference

This is a wrapper around a given [TNLP](#) class that takes out a list of constraints that are given to the constructor.

```
#include <IpTNLPReducer.hpp>
```

Inheritance diagram for Ipopt::TNLPReducer:



Public Member Functions

Constructors/Destructors

- [TNLPReducer](#) ([TNLP](#) &tnlp, [Index](#) n_g_skip, const [Index](#) *index_g_skip, [Index](#) n_xL_skip, const [Index](#) *index_xL_skip, [Index](#) n_xU_skip, const [Index](#) *index_xU_skip, [Index](#) n_x_fix, const [Index](#) *index_f_fix)
Constructor is given the indices of the constraints that should be taken out of the problem statement, as well as the original [TNLP](#).
- virtual [~TNLPReducer](#) ()
Default destructor.

Overloaded methods from TNLP

- virtual bool [get_nlp_info](#) ([Index](#) &n, [Index](#) &m, [Index](#) &nnz_jac_g, [Index](#) &nnz_h_lag, [IndexStyleEnum](#) &index_style)
- virtual bool [get_bounds_info](#) ([Index](#) n, [Number](#) *x_l, [Number](#) *x_u, [Index](#) m, [Number](#) *g_l, [Number](#) *g_u)
overload this method to return the information about the bound on the variables and constraints.
- virtual bool [get_scaling_parameters](#) ([Number](#) &obj_scaling, bool &use_x_scaling, [Index](#) n, [Number](#) *x_scaling, bool &use_g_scaling, [Index](#) m, [Number](#) *g_scaling)
overload this method to return scaling parameters.
- virtual bool [get_variables_linearity](#) ([Index](#) n, [LinearityType](#) *var_types)
overload this method to return the variables linearity ([TNLP::LINEAR](#) or [TNLP::NON_LINEAR](#)).
- virtual bool [get_constraints_linearity](#) ([Index](#) m, [LinearityType](#) *const_types)
overload this method to return the constraint linearity.
- virtual bool [get_starting_point](#) ([Index](#) n, bool init_x, [Number](#) *x, bool init_z, [Number](#) *z_L, [Number](#) *z_U, [Index](#) m, bool init_lambda, [Number](#) *lambda)
overload this method to return the starting point.
- virtual bool [get_warm_start_iterate](#) ([IteratesVector](#) &warm_start_iterate)
overload this method to provide an [Ipopt](#) iterate (already in the form [Ipopt](#) requires it internally) for a warm start.
- virtual bool [eval_f](#) ([Index](#) n, const [Number](#) *x, bool new_x, [Number](#) &obj_value)
overload this method to return the value of the objective function
- virtual bool [eval_grad_f](#) ([Index](#) n, const [Number](#) *x, bool new_x, [Number](#) *grad_f)
overload this method to return the vector of the gradient of the objective w.r.t.

- virtual bool `eval_g` (Index n, const Number *x, bool new_x, Index m, Number *g)
overload this method to return the vector of constraint values
- virtual bool `eval_jac_g` (Index n, const Number *x, bool new_x, Index m, Index nele_jac, Index *iRow, Index *jCol, Number *values)
overload this method to return the jacobian of the constraints.
- virtual bool `eval_h` (Index n, const Number *x, bool new_x, Number obj_factor, Index m, const Number *lambda, bool new_lambda, Index nele_hess, Index *iRow, Index *jCol, Number *values)
overload this method to return the hessian of the lagrangian.
- virtual void `finalize_solution` (SolverReturn status, Index n, const Number *x, const Number *z_L, const Number *z_U, Index m, const Number *g, const Number *lambda, Number obj_value, const IpoptData *ip_data, IpoptCalculatedQuantities *ip_cq)
*This method is called when the algorithm is complete so the **TNLP** can store/write the solution.*
- virtual bool `intermediate_callback` (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, const IpoptData *ip_data, IpoptCalculatedQuantities *ip_cq)
Intermediate Callback method for the user.
- virtual Index `get_number_of_nonlinear_variables` ()
- virtual bool `get_list_of_nonlinear_variables` (Index num_nonlin_vars, Index *pos_nonlin_vars)

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `TNLPReducer` ()
Default Constructor.
- `TNLPReducer` (const `TNLPReducer` &)
Copy Constructor.
- void `operator=` (const `TNLPReducer` &)
Overloaded Equals Operator.

Private Attributes

- Index `n_g_skip_`
Number of constraints to be skipped.
- Index * `index_g_skip_`
Array of indices of the constraints that are to be skipped.
- IndexStyleEnum `index_style_orig_`
Index style for original problem.
- Index * `g_keep_map_`
Map from original constraints to new constraints.
- Index `m_reduced_`
*Number of constraints in reduced **NLP**.*
- Index `nnz_jac_g_reduced_`
*Number of Jacobian nonzeros in the reduced **NLP**.*
- Index `nnz_jac_g_skipped_`
Number of Jacobian nonzeros that are skipped.
- Index * `jac_g_skipped_`

Array of Jacobian elements that are to be skipped.

- [Index n_xL_skip_](#)

Number of lower variable bounds to be skipped.

- [Index * index_xL_skip_](#)

Array of indices of the lower variable bounds to be skipped.

- [Index n_xU_skip_](#)

Number of upper variable bounds to be skipped.

- [Index * index_xU_skip_](#)

Array of indices of the upper variable bounds to be skipped.

- [Index n_x_fix_](#)

Number of variables that are to be fixed to initial value.

- [Index * index_x_fix_](#)

Array of indices of the variables that are to be fixed.

original TNLP

- [SmartPtr< TNLP > tnlp_](#)
- [Index m_orig_](#)
- [Index nnz_jac_g_orig_](#)

Additional Inherited Members

6.179.1 Detailed Description

This is a wrapper around a given [TNLP](#) class that takes out a list of constraints that are given to the constructor.

It is provided for convenience, if one wants to experiment with problems that consist of only a subset of the constraints. But keep in mind that this is not efficient, since behind the scenes we are still evaluation all functions and derivatives, and are making copies of the original data.

Definition at line 23 of file IpTNLPReducer.hpp.

6.179.2 Constructor & Destructor Documentation

6.179.2.1 `Ipopt::TNLPReducer::TNLPReducer (TNLP & tnlp, Index n_g_skip, const Index * index_g_skip, Index n_xL_skip, const Index * index_xL_skip, Index n_xU_skip, const Index * index_xU_skip, Index n_x_fix, const Index * index_f_fix)`

Constructor is given the indices of the constraints that should be taken out of the problem statement, as well as the original [TNLP](#).

6.179.2.2 `virtual Ipopt::TNLPReducer::~TNLPReducer () [virtual]`

Default destructor.

6.179.2.3 `Ipopt::TNLPReducer::TNLPReducer () [private]`

Default Constructor.

6.179.2.4 `Ipopt::TNLPReducer::TNLPReducer (const TNLPReducer &) [private]`

Copy Constructor.

6.179.3 Member Function Documentation

6.179.3.1 `virtual bool Ipopt::TNLPReducer::get_nlp_info (Index & n, Index & m, Index & nnz_jac_g, Index & nnz_h_lag, IndexStyleEnum & index_style) [virtual]`

Implements [Ipopt::TNLP](#).

6.179.3.2 `virtual bool Ipopt::TNLPReducer::get_bounds_info (Index n, Number * x_l, Number * x_u, Index m, Number * g_l, Number * g_u) [virtual]`

overload this method to return the information about the bound on the variables and constraints.

The value that indicates that a bound does not exist is specified in the parameters `nlp_lower_bound_inf` and `nlp_upper_bound_inf`. By default, `nlp_lower_bound_inf` is -1e19 and `nlp_upper_bound_inf` is 1e19. (see [TNLPAdapter](#))

Implements [Ipopt::TNLP](#).

6.179.3.3 `virtual bool Ipopt::TNLPReducer::get_scaling_parameters (Number & obj_scaling, bool & use_x_scaling, Index n, Number * x_scaling, bool & use_g_scaling, Index m, Number * g_scaling) [virtual]`

overload this method to return scaling parameters.

This is only called if the options are set to retrieve user scaling. There, `use_x_scaling` (or `use_g_scaling`) should get set to true only if the variables (or constraints) are to be scaled. This method should return true only if the scaling parameters could be provided.

Reimplemented from [Ipopt::TNLP](#).

6.179.3.4 `virtual bool Ipopt::TNLPReducer::get_variables_linearity (Index n, LinearityType * var_types) [virtual]`

overload this method to return the variables linearity ([TNLP::LINEAR](#) or [TNLP::NON_LINEAR](#)).

The `var_types` array has been allocated with length at least `n`. (default implementation just return false and does not fill the array).

Reimplemented from [Ipopt::TNLP](#).

6.179.3.5 `virtual bool Ipopt::TNLPReducer::get_constraints_linearity (Index m, LinearityType * const_types) [virtual]`

overload this method to return the constraint linearity.

array has been allocated with length at least `n`. (default implementation just return false and does not fill the array).

Reimplemented from [Ipopt::TNLP](#).

6.179.3.6 `virtual bool Ipopt::TNLPReducer::get_starting_point (Index n, bool init_x, Number * x, bool init_z, Number * z_L, Number * z_U, Index m, bool init_lambda, Number * lambda) [virtual]`

overload this method to return the starting point.

The bool variables indicate whether the algorithm wants you to initialize `x`, `z_L/z_u`, and `lambda`, respectively. If, for some reason, the algorithm wants you to initialize these and you cannot, return false, which will cause [Ipopt](#) to stop. You will have to run [Ipopt](#) with different options then.

Implements [Ipopt::TNLP](#).

6.179.3.7 `virtual bool Ipopt::TNLPReducer::get_warm_start_iterate (IteratesVector & warm_start_iterate) [virtual]`

overload this method to provide an [Ipopt](#) iterate (already in the form [Ipopt](#) requires it internally) for a warm start.

Since this is only for expert users, a default dummy implementation is provided and returns false.

Reimplemented from [Ipopt::TNLP](#).

6.179.3.8 `virtual bool Ipopt::TNLPReducer::eval_f (Index n, const Number * x, bool new_x, Number & obj_value)`
`[virtual]`

overload this method to return the value of the objective function

Implements [Ipopt::TNLP](#).

6.179.3.9 `virtual bool Ipopt::TNLPReducer::eval_grad_f (Index n, const Number * x, bool new_x, Number * grad_f)`
`[virtual]`

overload this method to return the vector of the gradient of the objective w.r.t.

x

Implements [Ipopt::TNLP](#).

6.179.3.10 `virtual bool Ipopt::TNLPReducer::eval_g (Index n, const Number * x, bool new_x, Index m, Number * g)`
`[virtual]`

overload this method to return the vector of constraint values

Implements [Ipopt::TNLP](#).

6.179.3.11 `virtual bool Ipopt::TNLPReducer::eval_jac_g (Index n, const Number * x, bool new_x, Index m, Index nele_jac,
Index * iRow, Index * jCol, Number * values)` `[virtual]`

overload this method to return the jacobian of the constraints.

The vectors *iRow* and *jCol* only need to be set once. The first call is used to set the structure only (*iRow* and *jCol* will be non-NULL, and values will be NULL) For subsequent calls, *iRow* and *jCol* will be NULL.

Implements [Ipopt::TNLP](#).

6.179.3.12 `virtual bool Ipopt::TNLPReducer::eval_h (Index n, const Number * x, bool new_x, Number obj_factor, Index m,
const Number * lambda, bool new_lambda, Index nele_hess, Index * iRow, Index * jCol, Number * values)`
`[virtual]`

overload this method to return the hessian of the lagrangian.

The vectors *iRow* and *jCol* only need to be set once (during the first call). The first call is used to set the structure only (*iRow* and *jCol* will be non-NULL, and values will be NULL) For subsequent calls, *iRow* and *jCol* will be NULL. This matrix is symmetric - specify the lower diagonal only. A default implementation is provided, in case the user wants to use quasi-Newton approximations to estimate the second derivatives and doesn't need to implement this method.

Reimplemented from [Ipopt::TNLP](#).

6.179.3.13 `virtual void Ipopt::TNLPReducer::finalize_solution (SolverReturn status, Index n, const Number * x, const
Number * z_L, const Number * z_U, Index m, const Number * g, const Number * lambda, Number obj_value,
const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq)` `[virtual]`

This method is called when the algorithm is complete so the [TNLP](#) can store/write the solution.

Implements [Ipopt::TNLP](#).

6.179.3.14 `virtual bool Ipopt::TNLPReducer::intermediate_callback (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, const IpoptData * ip_data, IpoptCalculatedQuantities * ip_cq)` `[virtual]`

Intermediate Callback method for the user.

Providing dummy default implementation. For details see IntermediateCallBack in [IpNLP.hpp](#).

Reimplemented from [Ipopt::TNLP](#).

6.179.3.15 `virtual Index Ipopt::TNLPReducer::get_number_of_nonlinear_variables ()` `[virtual]`

Reimplemented from [Ipopt::TNLP](#).

6.179.3.16 `virtual bool Ipopt::TNLPReducer::get_list_of_nonlinear_variables (Index num_nonlin_vars, Index * pos_nonlin_vars)` `[virtual]`

Reimplemented from [Ipopt::TNLP](#).

6.179.3.17 `void Ipopt::TNLPReducer::operator= (const TNLPReducer &)` `[private]`

Overloaded Equals Operator.

6.179.4 Member Data Documentation

6.179.4.1 `SmartPtr<TNLP> Ipopt::TNLPReducer::tnlp_` `[private]`

Definition at line 126 of file [IpTNLPReducer.hpp](#).

6.179.4.2 `Index Ipopt::TNLPReducer::m_orig_` `[private]`

Definition at line 127 of file [IpTNLPReducer.hpp](#).

6.179.4.3 `Index Ipopt::TNLPReducer::nnz_jac_g_orig_` `[private]`

Definition at line 128 of file [IpTNLPReducer.hpp](#).

6.179.4.4 `Index Ipopt::TNLPReducer::n_g_skip_` `[private]`

Number of constraints to be skipped.

Definition at line 132 of file [IpTNLPReducer.hpp](#).

6.179.4.5 `Index* Ipopt::TNLPReducer::index_g_skip_` `[private]`

Array of indices of the constraints that are to be skipped.

This is provided at the beginning in the constructor.

Definition at line 136 of file [IpTNLPReducer.hpp](#).

6.179.4.6 `IndexStyleEnum Ipopt::TNLPReducer::index_style_orig_` `[private]`

Index style for original problem.

Internally, we use C-Style now.

Definition at line 140 of file [IpTNLPReducer.hpp](#).

6.179.4.7 Index* Ipopt::TNLPReducer::g_keep_map_ [private]

Map from original constraints to new constraints.

A -1 means that a constraint is skipped.

Definition at line 144 of file IpTNLPReducer.hpp.

6.179.4.8 Index Ipopt::TNLPReducer::m_reduced_ [private]

Number of constraints in reduced [NLP](#).

Definition at line 147 of file IpTNLPReducer.hpp.

6.179.4.9 Index Ipopt::TNLPReducer::nnz_jac_g_reduced_ [private]

Number of Jacobian nonzeros in the reduced [NLP](#).

Definition at line 150 of file IpTNLPReducer.hpp.

6.179.4.10 Index Ipopt::TNLPReducer::nnz_jac_g_skipped_ [private]

Number of Jacobian nonzeros that are skipped.

Definition at line 153 of file IpTNLPReducer.hpp.

6.179.4.11 Index* Ipopt::TNLPReducer::jac_g_skipped_ [private]

Array of Jacobian elements that are to be skipped.

This is in increasing order.

Definition at line 157 of file IpTNLPReducer.hpp.

6.179.4.12 Index Ipopt::TNLPReducer::n_xL_skip_ [private]

Number of lower variable bounds to be skipped.

Definition at line 160 of file IpTNLPReducer.hpp.

6.179.4.13 Index* Ipopt::TNLPReducer::index_xL_skip_ [private]

Array of indices of the lower variable bounds to be skipped.

Definition at line 163 of file IpTNLPReducer.hpp.

6.179.4.14 Index Ipopt::TNLPReducer::n_xU_skip_ [private]

Number of upper variable bounds to be skipped.

Definition at line 166 of file IpTNLPReducer.hpp.

6.179.4.15 Index* Ipopt::TNLPReducer::index_xU_skip_ [private]

Array of indices of the upper variable bounds to be skipped.

Definition at line 169 of file IpTNLPReducer.hpp.

6.179.4.16 Index Ipopt::TNLPReducer::n_x_fix_ [private]

Number of variables that are to be fixed to initial value.

Definition at line 172 of file IpTNLPReducer.hpp.

6.179.4.17 `Index* Ipopt::TNLPReducer::index_x_fix_` [private]

Array of indices of the variables that are to be fixed.

Definition at line 175 of file `IpTNLPReducer.hpp`.

The documentation for this class was generated from the following file:

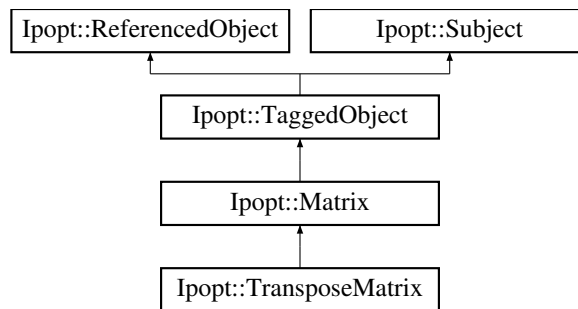
- [Interfaces/IpTNLPReducer.hpp](#)

6.180 Ipopt::TransposeMatrix Class Reference

Class for Matrices which are the transpose of another matrix.

```
#include <IpTransposeMatrix.hpp>
```

Inheritance diagram for `Ipopt::TransposeMatrix`:



Public Member Functions

Constructors / Destructors

- [TransposeMatrix](#) (const [TransposeMatrixSpace](#) *owner_space)
Constructor, initializing with dimensions of the matrix.
- [~TransposeMatrix](#) ()
Destructor.
- [SmartPtr](#)< const [Matrix](#) > [OrigMatrix](#) () const

Protected Member Functions

Methods overloaded from matrix

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual void [TransMultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix(transpose) vector multiply.
- virtual bool [IsValidNumbersImpl](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the columns in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TransposeMatrix](#) ()
Default Constructor.
- [TransposeMatrix](#) (const [TransposeMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [TransposeMatrix](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< [Matrix](#) > [orig_matrix_](#)
Pointer to original matrix.

Additional Inherited Members

6.180.1 Detailed Description

Class for Matrices which are the transpose of another matrix.

Definition at line 23 of file `IpTransposeMatrix.hpp`.

6.180.2 Constructor & Destructor Documentation

6.180.2.1 `Ipopt::TransposeMatrix::TransposeMatrix (const TransposeMatrixSpace * owner_space)`

Constructor, initializing with dimensions of the matrix.

6.180.2.2 `Ipopt::TransposeMatrix::~~TransposeMatrix () [inline]`

Destructor.

Definition at line 35 of file `IpTransposeMatrix.hpp`.

6.180.2.3 `Ipopt::TransposeMatrix::TransposeMatrix () [private]`

Default Constructor.

6.180.2.4 `Ipopt::TransposeMatrix::TransposeMatrix (const TransposeMatrix &) [private]`

Copy Constructor.

6.180.3 Member Function Documentation

6.180.3.1 `SmartPtr<const Matrix> Ipopt::TransposeMatrix::OrigMatrix () const [inline]`

Definition at line 38 of file `IpTransposeMatrix.hpp`.

6.180.3.2 `virtual void Ipopt::TransposeMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [inline], [protected], [virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

Definition at line 47 of file IpTransposeMatrix.hpp.

6.180.3.3 `virtual void Ipopt::TransposeMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [inline], [protected], [virtual]`

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements [Ipopt::Matrix](#).

Definition at line 54 of file IpTransposeMatrix.hpp.

6.180.3.4 `virtual bool Ipopt::TransposeMatrix::IsValidNumbersImpl () const [inline], [protected], [virtual]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Reimplemented from [Ipopt::Matrix](#).

Definition at line 63 of file IpTransposeMatrix.hpp.

6.180.3.5 `virtual void Ipopt::TransposeMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const [inline], [protected], [virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in *rows_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

Definition at line 69 of file IpTransposeMatrix.hpp.

6.180.3.6 `virtual void Ipopt::TransposeMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const [inline], [protected], [virtual]`

Compute the max-norm of the columns in the matrix.

The result is stored in *cols_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

Definition at line 75 of file IpTransposeMatrix.hpp.

6.180.3.7 `virtual void Ipopt::TransposeMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const [protected], [virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.180.3.8 `void Ipopt::TransposeMatrix::operator= (const TransposeMatrix &) [private]`

Overloaded Equals Operator.

6.180.4 Member Data Documentation

6.180.4.1 `SmartPointer<Matrix> Ipopt::TransposeMatrix::orig_matrix_` [private]

Pointer to original matrix.

Definition at line 109 of file `IpTransposeMatrix.hpp`.

The documentation for this class was generated from the following file:

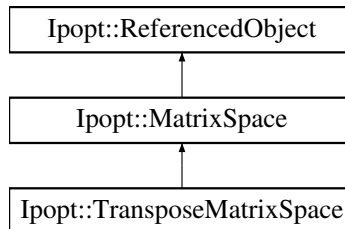
- [LinAlg/IpTransposeMatrix.hpp](#)

6.181 `Ipopt::TransposeMatrixSpace` Class Reference

This is the matrix space for [TransposeMatrix](#).

```
#include <IpTransposeMatrix.hpp>
```

Inheritance diagram for `Ipopt::TransposeMatrixSpace`:



Public Member Functions

- virtual `Matrix * MakeNew () const`
Overloaded MakeNew method for the [MatrixSpace](#) base class.
- `TransposeMatrix * MakeNewTransposeMatrix () const`
Method for creating a new matrix of this specific type.
- `Matrix * MakeNewOrigMatrix () const`

Constructors / Destructors

- `TransposeMatrixSpace (const MatrixSpace *orig_matrix_space)`
Constructor, given the dimension of the matrix.
- virtual `~TransposeMatrixSpace ()`
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `TransposeMatrixSpace ()`
Default Constructor.

- [TransposeMatrixSpace](#) (const [TransposeMatrixSpace](#) &)
Copy Constructor.
- void [operator=](#) (const [TransposeMatrixSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< const [MatrixSpace](#) > [orig_matrix_space_](#)
[Matrix](#) space of the original matrix.

6.181.1 Detailed Description

This is the matrix space for [TransposeMatrix](#).

Definition at line 113 of file IpTransposeMatrix.hpp.

6.181.2 Constructor & Destructor Documentation

6.181.2.1 `Ipopt::TransposeMatrixSpace::TransposeMatrixSpace (const MatrixSpace * orig_matrix_space) [inline]`

Constructor, given the dimension of the matrix.

Definition at line 119 of file IpTransposeMatrix.hpp.

6.181.2.2 `virtual Ipopt::TransposeMatrixSpace::~~TransposeMatrixSpace () [inline],[virtual]`

Destructor.

Definition at line 126 of file IpTransposeMatrix.hpp.

6.181.2.3 `Ipopt::TransposeMatrixSpace::TransposeMatrixSpace () [private]`

Default Constructor.

6.181.2.4 `Ipopt::TransposeMatrixSpace::TransposeMatrixSpace (const TransposeMatrixSpace &) [private]`

Copy Constructor.

6.181.3 Member Function Documentation

6.181.3.1 `virtual Matrix* Ipopt::TransposeMatrixSpace::MakeNew () const [inline],[virtual]`

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 132 of file IpTransposeMatrix.hpp.

6.181.3.2 `TransposeMatrix* Ipopt::TransposeMatrixSpace::MakeNewTransposeMatrix () const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 138 of file IpTransposeMatrix.hpp.

6.181.3.3 **Matrix*** **Ipopt::TransposeMatrixSpace::MakeNewOrigMatrix () const** `[inline]`

Definition at line 143 of file `IpTransposeMatrix.hpp`.

6.181.3.4 **void** **Ipopt::TransposeMatrixSpace::operator= (const TransposeMatrixSpace &)** `[private]`

Overloaded Equals Operator.

6.181.4 Member Data Documentation

6.181.4.1 **SmartPtr<const MatrixSpace>** **Ipopt::TransposeMatrixSpace::orig_matrix_space_** `[private]`

[Matrix](#) space of the original matrix.

Definition at line 168 of file `IpTransposeMatrix.hpp`.

The documentation for this class was generated from the following file:

- [LinAlg/IpTransposeMatrix.hpp](#)

6.182 Ipopt::TripletToCSRConverter::TripletEntry Class Reference

Class for one triplet position entry.

Public Member Functions

- **void** **Set** ([Index](#) i_row, [Index](#) j_col, [Index](#) i_pos_triplet)
Set the values of an entry.
- **bool** **operator<** (const [TripletEntry](#) &Tentry) const
Comparison operator.

Constructor/Destructor

- [TripletEntry](#) ()
Constructor.
- [~TripletEntry](#) ()
Destructor.
- [TripletEntry](#) (const [TripletEntry](#) &rhs)
Copy constructor, required for std::list.
- [TripletEntry](#) & **operator=** (const [TripletEntry](#) &rhs)
Equals Operator, required for std::list.

Accessor methods.

- [Index](#) **IRow** () const
Row position.
- [Index](#) **JCol** () const
Column position.
- [Index](#) **PosTriplet** () const
Index in original triplet matrix.

Private Attributes

Entry content.*Default Constructor**Copy Constructor*

- [Index i_row_](#)
- [Index j_col_](#)
- [Index i_pos_triplet_](#)

6.182.1 Detailed Description

Class for one triplet position entry.

Definition at line 26 of file IpTripletToCSRConverter.hpp.

6.182.2 Constructor & Destructor Documentation

6.182.2.1 Ipopt::TripletToCSRConverter::TripletEntry () [\[inline\]](#)

Constructor.

Definition at line 32 of file IpTripletToCSRConverter.hpp.

6.182.2.2 Ipopt::TripletToCSRConverter::TripletEntry::~~TripletEntry () [\[inline\]](#)

Destructor.

Definition at line 36 of file IpTripletToCSRConverter.hpp.

6.182.2.3 Ipopt::TripletToCSRConverter::TripletEntry::TripletEntry (const TripletEntry & rhs) [\[inline\]](#)

Copy constructor, required for std::list.

Definition at line 40 of file IpTripletToCSRConverter.hpp.

6.182.3 Member Function Documentation

6.182.3.1 TripletEntry& Ipopt::TripletToCSRConverter::TripletEntry::operator= (const TripletEntry & rhs) [\[inline\]](#)

Equals Operator, required for std::list.

Definition at line 48 of file IpTripletToCSRConverter.hpp.

6.182.3.2 void Ipopt::TripletToCSRConverter::TripletEntry::Set (Index i_row, Index j_col, Index i_pos_triplet) [\[inline\]](#)

Set the values of an entry.

Definition at line 60 of file IpTripletToCSRConverter.hpp.

6.182.3.3 Index Ipopt::TripletToCSRConverter::TripletEntry::IRow () const [\[inline\]](#)

Row position.

Definition at line 76 of file IpTripletToCSRConverter.hpp.

6.182.3.4 Index `Ipopt::TripletToCSRConverter::TripletEntry::JCol () const` [inline]

Column position.

Definition at line 81 of file `IpTripletToCSRConverter.hpp`.

6.182.3.5 Index `Ipopt::TripletToCSRConverter::TripletEntry::PosTriplet () const` [inline]

Index in original triplet matrix.

Definition at line 86 of file `IpTripletToCSRConverter.hpp`.

6.182.3.6 `bool Ipopt::TripletToCSRConverter::TripletEntry::operator< (const TripletEntry & Tentry) const` [inline]

Comparison operator.

This is required for the sort function.

Definition at line 93 of file `IpTripletToCSRConverter.hpp`.

6.182.4 Member Data Documentation

6.182.4.1 Index `Ipopt::TripletToCSRConverter::TripletEntry::i_row_` [private]

Definition at line 119 of file `IpTripletToCSRConverter.hpp`.

6.182.4.2 Index `Ipopt::TripletToCSRConverter::TripletEntry::j_col_` [private]

Definition at line 120 of file `IpTripletToCSRConverter.hpp`.

6.182.4.3 Index `Ipopt::TripletToCSRConverter::TripletEntry::i_pos_triplet_` [private]

Definition at line 121 of file `IpTripletToCSRConverter.hpp`.

The documentation for this class was generated from the following file:

- Algorithm/LinearSolvers/[IpTripletToCSRConverter.hpp](#)

6.183 Ipopt::TripletHelper Class Reference

```
#include <IpTripletHelper.hpp>
```

Static Public Member Functions

A set of recursive routines that help with the Triplet format.

- static [Index](#) `GetNumberEntries` (const [Matrix](#) &matrix)
find the total number of triplet entries of a [Matrix](#)
- static void `FillRowCol` ([Index](#) n_entries, const [Matrix](#) &matrix, [Index](#) *iRow, [Index](#) *jCol, [Index](#) row_offset=0, [Index](#) col_offset=0)
fill the irows, jcols structure for the triplet format from the matrix
- static void `FillValues` ([Index](#) n_entries, const [Matrix](#) &matrix, [Number](#) *values)
fill the values for the triplet format from the matrix
- static void `FillValuesFromVector` ([Index](#) dim, const [Vector](#) &vector, [Number](#) *values)
fill the values from the vector into a dense double structure*
- static void `PutValuesInVector` ([Index](#) dim, const double *values, [Vector](#) &vector)
put the values from the double back into the vector*

Static Private Member Functions

- static [Index GetNumberEntries_](#) (const [SumMatrix](#) &matrix)
find the total number of triplet entries for the [SumMatrix](#)
- static [Index GetNumberEntries_](#) (const [SumSymMatrix](#) &matrix)
find the total number of triplet entries for the [SumSymMatrix](#)
- static [Index GetNumberEntries_](#) (const [CompoundMatrix](#) &matrix)
find the total number of triplet entries for the [CompoundMatrix](#)
- static [Index GetNumberEntries_](#) (const [CompoundSymMatrix](#) &matrix)
find the total number of triplet entries for the [CompoundSymMatrix](#)
- static [Index GetNumberEntries_](#) (const [TransposeMatrix](#) &matrix)
find the total number of triplet entries for the [TransposeMatrix](#)
- static [Index GetNumberEntries_](#) (const [ExpandedMultiVectorMatrix](#) &matrix)
find the total number of triplet entries for the [TransposeMatrix](#)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [GenTMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [GenTMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [SymTMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [SymTMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [DiagMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [DiagMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [IdentityMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [IdentityMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [ExpansionMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [ExpansionMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [SumMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [SumMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [SumSymMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [SumSymMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [CompoundMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [CompoundMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [CompoundSymMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [CompoundSymMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [ScaledMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [ScaledMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [SymScaledMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [SymScaledMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [TransposeMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [TransposeMatrix](#) &matrix, [Number](#) *values)
- static void [FillRowCol_](#) ([Index](#) n_entries, const [ExpandedMultiVectorMatrix](#) &matrix, [Index](#) row_offset, [Index](#) col_offset, [Index](#) *iRow, [Index](#) *jCol)
- static void [FillValues_](#) ([Index](#) n_entries, const [ExpandedMultiVectorMatrix](#) &matrix, [Number](#) *values)

6.183.1 Detailed Description

Definition at line 40 of file IpTripletHelper.hpp.

6.183.2 Member Function Documentation

6.183.2.1 `static Index Ipopt::TripletHelper::GetNumberEntries (const Matrix & matrix) [static]`

find the total number of triplet entries of a [Matrix](#)

6.183.2.2 `static void Ipopt::TripletHelper::FillRowCol (Index n_entries, const Matrix & matrix, Index * iRow, Index * jCol, Index row_offset = 0, Index col_offset = 0) [static]`

fill the irows, jcols structure for the triplet format from the matrix

6.183.2.3 `static void Ipopt::TripletHelper::FillValues (Index n_entries, const Matrix & matrix, Number * values) [static]`

fill the values for the triplet format from the matrix

6.183.2.4 `static void Ipopt::TripletHelper::FillValuesFromVector (Index dim, const Vector & vector, Number * values) [static]`

fill the values from the vector into a dense double* structure

6.183.2.5 `static void Ipopt::TripletHelper::PutValuesInVector (Index dim, const double * values, Vector & vector) [static]`

put the values from the double* back into the vector

6.183.2.6 `static Index Ipopt::TripletHelper::GetNumberEntries_ (const SumMatrix & matrix) [static], [private]`

find the total number of triplet entries for the [SumMatrix](#)

6.183.2.7 `static Index Ipopt::TripletHelper::GetNumberEntries_ (const SumSymMatrix & matrix) [static], [private]`

find the total number of triplet entries for the [SumSymMatrix](#)

6.183.2.8 `static Index Ipopt::TripletHelper::GetNumberEntries_ (const CompoundMatrix & matrix) [static], [private]`

find the total number of triplet entries for the [CompoundMatrix](#)

6.183.2.9 `static Index Ipopt::TripletHelper::GetNumberEntries_ (const CompoundSymMatrix & matrix) [static], [private]`

find the total number of triplet entries for the [CompoundSymMatrix](#)

6.183.2.10 `static Index Ipopt::TripletHelper::GetNumberEntries_ (const TransposeMatrix & matrix) [static], [private]`

find the total number of triplet entries for the [TransposeMatrix](#)

6.183.2.11 `static Index Ipopt::TripletHelper::GetNumberEntries_ (const ExpandedMultiVectorMatrix & matrix) [static], [private]`

find the total number of triplet entries for the [TransposeMatrix](#)

- 6.183.2.12 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const GenTMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.13 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const GenTMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.14 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const SymTMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.15 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const SymTMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.16 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const DiagMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.17 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const DiagMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.18 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const IdentityMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.19 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const IdentityMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.20 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const ExpansionMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.21 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const ExpansionMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.22 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const SumMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.23 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const SumMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.24 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const SumSymMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.25 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const SumSymMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.26 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const CompoundMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.27 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const CompoundMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.28 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const CompoundSymMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]
- 6.183.2.29 static void Ipopt::TripletHelper::FillValues_ (Index *n_entries*, const CompoundSymMatrix & *matrix*, Number * *values*) [static], [private]
- 6.183.2.30 static void Ipopt::TripletHelper::FillRowCol_ (Index *n_entries*, const ScaledMatrix & *matrix*, Index *row_offset*, Index *col_offset*, Index * *iRow*, Index * *jCol*) [static], [private]

- 6.183.2.31 `static void Ipopt::TripletHelper::FillValues_ (Index n_entries, const ScaledMatrix & matrix, Number * values)`
[static], [private]
- 6.183.2.32 `static void Ipopt::TripletHelper::FillRowCol_ (Index n_entries, const SymScaledMatrix & matrix, Index row_offset, Index col_offset, Index * iRow, Index * jCol)` [static], [private]
- 6.183.2.33 `static void Ipopt::TripletHelper::FillValues_ (Index n_entries, const SymScaledMatrix & matrix, Number * values)`
[static], [private]
- 6.183.2.34 `static void Ipopt::TripletHelper::FillRowCol_ (Index n_entries, const TransposeMatrix & matrix, Index row_offset, Index col_offset, Index * iRow, Index * jCol)` [static], [private]
- 6.183.2.35 `static void Ipopt::TripletHelper::FillValues_ (Index n_entries, const TransposeMatrix & matrix, Number * values)`
[static], [private]
- 6.183.2.36 `static void Ipopt::TripletHelper::FillRowCol_ (Index n_entries, const ExpandedMultiVectorMatrix & matrix, Index row_offset, Index col_offset, Index * iRow, Index * jCol)` [static], [private]
- 6.183.2.37 `static void Ipopt::TripletHelper::FillValues_ (Index n_entries, const ExpandedMultiVectorMatrix & matrix, Number * values)` [static], [private]

The documentation for this class was generated from the following file:

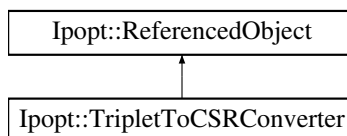
- [LinAlg/TMatrices/lpTripletHelper.hpp](#)

6.184 Ipopt::TripletToCSRConverter Class Reference

Class for converting symmetric matrices given in triplet format to matrices in compressed sparse row (CSR) format of the upper triangular part (or, equivalently, compressed sparse column (CSC) format for the lower triangular part).

```
#include <IpTripletToCSRConverter.hpp>
```

Inheritance diagram for Ipopt::TripletToCSRConverter:



Classes

- class [TripletEntry](#)
Class for one triplet position entry.

Public Types

- enum [ETriFull](#) { [Triangular_Format](#), [Full_Format](#) }
Enum to specify half or full matrix storage.

Public Member Functions

- [Index InitializeConverter](#) (Index dim, Index nonzeros, const Index *airn, const Index *ajcn)

Initialize the converter, given the fixed structure of the matrix.

- void [ConvertValues](#) ([Index](#) nonzeros_triplet, const [Number](#) *a_triplet, [Index](#) nonzeros_compressed, [Number](#) *a_compressed)

Convert the values of the nonzero elements.

Constructor/Destructor

- [TripletToCSRConverter](#) ([Index](#) offset, [ETriFull](#) hf=[Triangular_Format](#))
- virtual [~TripletToCSRConverter](#) ()

Destructor.

Accessor methods

- const [Index](#) * [IA](#) () const
Return the IA array for the condensed format.
- const [Index](#) * [JA](#) () const
Return the JA array for the condensed format.
- const [Index](#) * [iPosFirst](#) () const

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TripletToCSRConverter](#) ()
Default Constructor.
- [TripletToCSRConverter](#) (const [TripletToCSRConverter](#) &)
Copy Constructor.
- void [operator=](#) (const [TripletToCSRConverter](#) &)
Overloaded Equals Operator.

Private Attributes

- [Index](#) [offset_](#)
Offset for CSR numbering.
- [ETriFull](#) [hf_](#)
Indicator of half (ie lower only) or full (both upr and lwr) matrix.
- [Index](#) * [ia_](#)
Array storing the values for IA in the condensed format.
- [Index](#) * [ja_](#)
Array storing the values for JA in the condensed format.
- [Index](#) [dim_](#)
Dimension of the matrix.
- [Index](#) [nonzeros_triplet_](#)
Number of nonzeros in the triplet format.
- [Index](#) [nonzeros_compressed_](#)
Number of nonzeros in the compressed format.
- [Index](#) [num_doubles_](#)

Number of repeated entries.

- bool `initialized_`

Flag indicating if initialize method had been called.

Arrays for cross-positions for the conversion of values.

- Index * `ipos_first_`

First elements assignment.

- Index * `ipos_double_triplet_`

Position of multiple elements in triplet matrix.

- Index * `ipos_double_compressed_`

Position of multiple elements in compressed matrix.

6.184.1 Detailed Description

Class for converting symmetric matrices given in triplet format to matrices in compressed sparse row (CSR) format of the upper triangular part (or, equivalently, compressed sparse column (CSC) format for the lower triangular part).

In the description for this class, we assume that we discuss the CSR format.

Definition at line 23 of file `IpTripletToCSRConverter.hpp`.

6.184.2 Member Enumeration Documentation

6.184.2.1 enum `Ipopt::TripletToCSRConverter::ETriFull`

Enum to specify half or full matrix storage.

Enumerator

Triangular_Format Lower (or Upper) triangular stored only.

Full_Format Store both lower and upper parts.

Definition at line 127 of file `IpTripletToCSRConverter.hpp`.

6.184.3 Constructor & Destructor Documentation

6.184.3.1 `Ipopt::TripletToCSRConverter::TripletToCSRConverter (Index offset, ETriFull hf = Triangular_Format)`

6.184.3.2 `virtual Ipopt::TripletToCSRConverter::~~TripletToCSRConverter () [virtual]`

Destructor.

6.184.3.3 `Ipopt::TripletToCSRConverter::TripletToCSRConverter () [private]`

Default Constructor.

6.184.3.4 `Ipopt::TripletToCSRConverter::TripletToCSRConverter (const TripletToCSRConverter &) [private]`

Copy Constructor.

6.184.4 Member Function Documentation

6.184.4.1 Index Ipopt::TripletToCSRConverter::InitializeConverter (Index *dim*, Index *nonzeros*, const Index * *airn*, const Index * *ajcn*)

Initialize the converter, given the fixed structure of the matrix.

There, *ndim* gives the number of rows and columns of the matrix, *nonzeros* give the number of nonzero elements, and *airn* and *ajcn* give the positions of the nonzero elements. The return value is the number of nonzeros in the condensed matrix. (Since nonzero elements can be listed several times in the triplet format, it is possible that this value is different from the input value *nonzeros*.) This method must be called before the *GetIA*, *GetJA*, *Convert Values* methods are called.

6.184.4.2 const Index* Ipopt::TripletToCSRConverter::IA () const [inline]

Return the IA array for the condensed format.

Definition at line 163 of file *IpTripletToCSRConverter.hpp*.

6.184.4.3 const Index* Ipopt::TripletToCSRConverter::JA () const [inline]

Return the JA array for the condensed format.

Definition at line 170 of file *IpTripletToCSRConverter.hpp*.

6.184.4.4 const Index* Ipopt::TripletToCSRConverter::iPosFirst () const [inline]

Definition at line 175 of file *IpTripletToCSRConverter.hpp*.

6.184.4.5 void Ipopt::TripletToCSRConverter::ConvertValues (Index *nonzeros_triplet*, const Number * *a_triplet*, Index *nonzeros_compressed*, Number * *a_compressed*)

Convert the values of the nonzero elements.

Given the values *a_triplet* for the triplet format, return the array of values for the condensed format in *a_compressed*. *nonzeros_compressed* is the length of the array *a_compressed* and must be identical to the return value of *InitializeConverter*.

6.184.4.6 void Ipopt::TripletToCSRConverter::operator= (const TripletToCSRConverter &) [private]

Overloaded Equals Operator.

6.184.5 Member Data Documentation

6.184.5.1 Index Ipopt::TripletToCSRConverter::offset_ [private]

Offset for CSR numbering.

Definition at line 210 of file *IpTripletToCSRConverter.hpp*.

6.184.5.2 ETriFull Ipopt::TripletToCSRConverter::hf_ [private]

Indicator of half (ie lower only) or full (both upr and lwr) matrix.

Definition at line 213 of file *IpTripletToCSRConverter.hpp*.

6.184.5.3 Index* Ipopt::TripletToCSRConverter::ia_ [private]

Array storing the values for IA in the condensed format.

Definition at line 216 of file IpTripletToCSRConverter.hpp.

6.184.5.4 Index* Ipopt::TripletToCSRConverter::ja_ [private]

Array storing the values for JA in the condensed format.

Definition at line 219 of file IpTripletToCSRConverter.hpp.

6.184.5.5 Index Ipopt::TripletToCSRConverter::dim_ [private]

Dimension of the matrix.

Definition at line 222 of file IpTripletToCSRConverter.hpp.

6.184.5.6 Index Ipopt::TripletToCSRConverter::nonzeros_triplet_ [private]

Number of nonzeros in the triplet format.

Definition at line 225 of file IpTripletToCSRConverter.hpp.

6.184.5.7 Index Ipopt::TripletToCSRConverter::nonzeros_compressed_ [private]

Number of nonzeros in the compressed format.

Definition at line 228 of file IpTripletToCSRConverter.hpp.

6.184.5.8 Index Ipopt::TripletToCSRConverter::num_doubles_ [private]

Number of repeated entries.

Definition at line 231 of file IpTripletToCSRConverter.hpp.

6.184.5.9 bool Ipopt::TripletToCSRConverter::initialized_ [private]

Flag indicating if initialize method had been called.

Definition at line 234 of file IpTripletToCSRConverter.hpp.

6.184.5.10 Index* Ipopt::TripletToCSRConverter::ipos_first_ [private]

First elements assignement.

For i with $0 \leq i \leq \text{nonzeros_compressed}-1$, the i -th element in the compressed format is obtained from copying the $\text{ipos_filter}[i]$ -th element from the triplet format.

Definition at line 242 of file IpTripletToCSRConverter.hpp.

6.184.5.11 Index* Ipopt::TripletToCSRConverter::ipos_double_triplet_ [private]

Position of multiple elements in triplet matrix.

For $i = 0, \dots, \text{nonzeros_triplet}-\text{nonzeros_compressed}$, the $\text{ipos_double_triplet}[i]$ -th element in the triplet matrix has to be added to the $\text{ipos_double_compressed}[i]$ -th element in the compressed matrix.

Definition at line 248 of file IpTripletToCSRConverter.hpp.

6.184.5.12 Index* Ipopt::TripletToCSRConverter::ipos_double_compressed_ [private]

Position of multiple elements in compressed matrix.

Definition at line 250 of file IpTripletToCSRConverter.hpp.

The documentation for this class was generated from the following file:

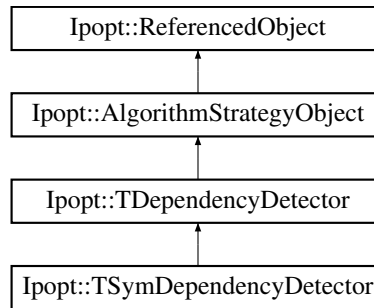
- Algorithm/LinearSolvers/[IpTripletToCSRConverter.hpp](#)

6.185 Ipopt::TSymDependencyDetector Class Reference

Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

```
#include <IpTSymDependencyDetector.hpp>
```

Inheritance diagram for Ipopt::TSymDependencyDetector:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
Has to be called to initialize and reset these objects.
- virtual bool [DetermineDependentRows](#) ([Index](#) n_rows, [Index](#) n_cols, [Index](#) n_jac_nz, [Number](#) *jac_c_vals, [Index](#) *jac_c_iRow, [Index](#) *jac_c_jCol, std::list< [Index](#) > &c_deps)
Method determining the number of linearly dependent rows in the matrix and the indices of those rows.

Constructor/Destructor

- [TSymDependencyDetector](#) ([TSymLinearSolver](#) &tsym_linear_solver)
- virtual [~TSymDependencyDetector](#) ()

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
This must be called to make the options for this class known.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TSymDependencyDetector](#) ()
Default Constructor.
- [TSymDependencyDetector](#) (const [TSymDependencyDetector](#) &)
Copy Constructor.
- void [operator=](#) (const [TSymDependencyDetector](#) &)
Overloaded Equals Operator.

Private Attributes

- [SmartPtr](#)< const [Journalist](#) > [jnlst_](#)
- [SmartPtr](#)< [TSymLinearSolver](#) > [tsym_linear_solver_](#)

Additional Inherited Members

6.185.1 Detailed Description

Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

Definition at line 20 of file `IpTSymDependencyDetector.hpp`.

6.185.2 Constructor & Destructor Documentation

6.185.2.1 `Ipopt::TSymDependencyDetector::TSymDependencyDetector (TSymLinearSolver & tsym_linear_solver)`

6.185.2.2 `virtual Ipopt::TSymDependencyDetector::~~TSymDependencyDetector ()` `[inline]`, `[virtual]`

Definition at line 27 of file `IpTSymDependencyDetector.hpp`.

6.185.2.3 `Ipopt::TSymDependencyDetector::TSymDependencyDetector ()` `[private]`

Default Constructor.

6.185.2.4 `Ipopt::TSymDependencyDetector::TSymDependencyDetector (const TSymDependencyDetector &)`
`[private]`

Copy Constructor.

6.185.3 Member Function Documentation

6.185.3.1 `virtual bool Ipopt::TSymDependencyDetector::InitializeImpl (const OptionsList & options, const std::string & prefix)`
`[virtual]`

Has to be called to initialize and reset these objects.

Implements [Ipopt::TDependencyDetector](#).

6.185.3.2 `virtual bool Ipopt::TSymDependencyDetector::DetermineDependentRows (Index n_rows, Index n_cols, Index n_jac_nz, Number * jac_c_vals, Index * jac_c_iRow, Index * jac_c_jCol, std::list< Index > & c_deps)`
`[virtual]`

Method determining the number of linearly dependent rows in the matrix and the indices of those rows.

We assume that the matrix is available in "Triplet" format (MA28 format), and that the arrays given to this method can be modified internally, i.e., they are not used by the calling program anymore after this call. This method returns false if there was a problem with the underlying linear solver.

Implements [Ipopt::TDependencyDetector](#).

6.185.3.3 `static void Ipopt::TSymDependencyDetector::RegisterOptions (SmartPtr< RegisteredOptions > roptions)`
`[static]`

This must be called to make the options for this class known.

6.185.3.4 `void Ipopt::TSymDependencyDetector::operator=(const TSymDependencyDetector &) [private]`

Overloaded Equals Operator.

6.185.4 Member Data Documentation

6.185.4.1 `SmartPtr<const Journalist> Ipopt::TSymDependencyDetector::jnlst_ [private]`

Definition at line 73 of file IpTSymDependencyDetector.hpp.

6.185.4.2 `SmartPtr<TSymLinearSolver> Ipopt::TSymDependencyDetector::tsym_linear_solver_ [private]`

Definition at line 75 of file IpTSymDependencyDetector.hpp.

The documentation for this class was generated from the following file:

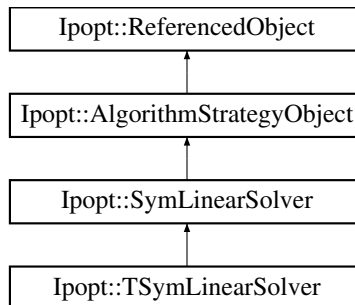
- [Algorithm/LinearSolvers/IpTSymDependencyDetector.hpp](#)

6.186 Ipopt::TSymLinearSolver Class Reference

General driver for linear solvers for sparse indefinite symmetric matrices.

```
#include <IpTSymLinearSolver.hpp>
```

Inheritance diagram for Ipopt::TSymLinearSolver:



Public Member Functions

- `bool InitializeImpl (const OptionsList &options, const std::string &prefix)`
overloaded from [AlgorithmStrategyObject](#)

Constructor/Destructor

- `TSymLinearSolver (SmartPtr< SparseSymLinearSolverInterface > solver_interface, SmartPtr< TSymScalingMethod > scaling_method)`
Constructor.
- `virtual ~TSymLinearSolver ()`
Destructor.

Methods for requesting solution of the linear system.

- `virtual ESymSolverStatus MultiSolve (const SymMatrix &A, std::vector< SmartPtr< const Vector > > &rhsV, std::vector< SmartPtr< Vector > > &solV, bool check_NegEVals, Index numberOfNegEVals)`

Solve operation for multiple right hand sides.

- virtual [Index NumberOfNegEvals](#) () const

Number of negative eigenvalues detected during last factorization.

- virtual bool [IncreaseQuality](#) ()

Request to increase quality of solution for next solve.

- virtual bool [ProvidesInertia](#) () const

Query whether inertia is computed by linear solver.

Methods related to the detection of linearly dependent

rows in a matrix

- bool [ProvidesDegeneracyDetection](#) () const

Returns true if the underlying linear solver can detect the linearly dependent rows in a matrix.

- [ESymSolverStatus DetermineDependentRows](#) ([Index](#) n_rows, [Index](#) n_cols, [Index](#) n_jac_nz, [Number](#) *jac_c_vals, [Index](#) *jac_c_iRow, [Index](#) *jac_c_jCol, std::list< [Index](#) > &c_deps)

Given the entries of a matrix in Triplet format, this method determines the list of row indices of the linearly dependent rows.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)

Methods for [OptionsList](#).

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TSymLinearSolver](#) ()
Default Constructor.
- [TSymLinearSolver](#) (const [TSymLinearSolver](#) &)
Copy Constructor.
- void [operator=](#) (const [TSymLinearSolver](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus InitializeStructure](#) (const [SymMatrix](#) &symT_A)
Initialize nonzero structure.
- void [GiveMatrixToSolver](#) (bool new_matrix, const [SymMatrix](#) &sym_A)
Copy the elements of the matrix in the required format into the array that is provided by the solver interface.

Private Attributes

- [SmartPtr](#)
< [SparseSymLinearSolverInterface](#) > [solver_interface_](#)
Strategy Object for an interface to a linear solver.

Information about the matrix

- [TaggedObject::Tag](#) [atag_](#)
Tag for the incoming matrix.
- [Index](#) [dim_](#)
Number of rows and columns of the matrix.
- [Index](#) [nonzeros_triplet_](#)
Number of nonzeros of the matrix in triplet format.
- [Index](#) [nonzeros_compressed_](#)
Number of nonzeros in compressed format.

Initialization flags

- bool [have_structure_](#)
Flag indicating if the internal structures are initialized.
- bool [linear_scaling_on_demand_](#)
Flag indicating whether the scaling objected is to be switched on when increased quality is requested.
- bool [initialized_](#)
Flag indicating if the InitializeStructure method has been called for the linear solver.

Stuff for scaling of the linear system.

- [SmartPtr](#) < [TSymScalingMethod](#) > [scaling_method_](#)
Strategy Object for a method that computes scaling factors for the matrices.
- double * [scaling_factors_](#)
Array storing the scaling factors.
- bool [use_scaling_](#)
Flag indicating whether scaling should be performed.
- bool [just_switched_on_scaling_](#)
Flag indicating whether we just switched on the scaling.

information about the matrix.

- [Index](#) * [airn_](#)
row indices of matrix in triplet (MA27) format.
- [Index](#) * [ajcn_](#)
column indices of matrix in triplet (MA27) format.
- [SmartPtr](#) < [TripletToCSRConverter](#) > [triplet_to_csr_converter_](#)
Pointer to object for conversion from triplet to compressed format.
- [SparseSymLinearSolverInterface::EMatrixFormat](#) [matrix_format_](#)
Flag indicating what matrix data format the solver requires.

Algorithmic parameters

- bool [warm_start_same_structure_](#)
Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Additional Inherited Members

6.186.1 Detailed Description

General driver for linear solvers for sparse indefinite symmetric matrices.

This interface includes a call to a method for scaling of the matrix (if given). This class takes in the constructor a pointer to the interface to an actual linear solver, and possibly a pointer to a method for computing scaling factors. It translates the [SymMatrix](#) into the format required by the linear solver and calls the solver via the [TSymLinearSolverInterface](#). If a scaling method has been given, the matrix, the right hand side, and the solution are scaled.

Definition at line 33 of file [IpTSymLinearSolver.hpp](#).

6.186.2 Constructor & Destructor Documentation

6.186.2.1 `Ipopt::TSymLinearSolver::TSymLinearSolver (SmartPtr< SparseSymLinearSolverInterface > solver_interface, SmartPtr< TSymScalingMethod > scaling_method)`

Constructor.

The `solver_interface` is a pointer to a linear solver for symmetric matrices in triplet format. If `scaling_method` not NULL, it must be a pointer to a class for computing scaling factors for the matrix.

6.186.2.2 `virtual Ipopt::TSymLinearSolver::~~TSymLinearSolver () [virtual]`

Destructor.

6.186.2.3 `Ipopt::TSymLinearSolver::TSymLinearSolver () [private]`

Default Constructor.

6.186.2.4 `Ipopt::TSymLinearSolver::TSymLinearSolver (const TSymLinearSolver &) [private]`

Copy Constructor.

6.186.3 Member Function Documentation

6.186.3.1 `bool Ipopt::TSymLinearSolver::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SymLinearSolver](#).

6.186.3.2 `virtual ESymSolverStatus Ipopt::TSymLinearSolver::MultiSolve (const SymMatrix & A, std::vector< SmartPtr< const Vector > > & rhsV, std::vector< SmartPtr< Vector > > & solV, bool check_NegEVals, Index numberOfNegEVals) [virtual]`

Solve operation for multiple right hand sides.

For details see the description in the base class [SymLinearSolver](#).

Implements [Ipopt::SymLinearSolver](#).

6.186.3.3 `virtual Index Ipopt::TSymLinearSolver::NumberOfNegEVals () const [virtual]`

Number of negative eigenvalues detected during last factorization.

Returns the number of negative eigenvalues of the most recent factorized matrix.

Implements [Ipopt::SymLinearSolver](#).

6.186.3.4 `virtual bool Ipopt::TSymLinearSolver::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

Ask linear solver to increase quality of solution for the next solve (e.g. increase pivot tolerance). Returns false, if this is not possible (e.g. maximal pivot tolerance already used.)

Implements [Ipopt::SymLinearSolver](#).

6.186.3.5 `virtual bool Ipopt::TSymLinearSolver::ProvidesInertia () const [virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SymLinearSolver](#).

6.186.3.6 `bool Ipopt::TSymLinearSolver::ProvidesDegeneracyDetection () const`

Returns true if the underlying linear solver can detect the linearly dependent rows in a matrix.

6.186.3.7 `ESymSolverStatus Ipopt::TSymLinearSolver::DetermineDependentRows (Index n_rows, Index n_cols, Index n_jac_nz, Number * jac_c_vals, Index * jac_c_iRow, Index * jac_c_jCol, std::list< Index > & c_deps)`

Given the entries of a matrix in Triplet format, this method determines the list of row indices of the linearly dependent rows.

This is a specific implementation for Triplet matrices.

6.186.3.8 `static void Ipopt::TSymLinearSolver::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]`

Methods for [OptionsList](#).

6.186.3.9 `void Ipopt::TSymLinearSolver::operator= (const TSymLinearSolver &) [private]`

Overloaded Equals Operator.

6.186.3.10 `ESymSolverStatus Ipopt::TSymLinearSolver::InitializeStructure (const SymMatrix & symT_A) [private]`

Initialize nonzero structure.

Set dim_ and nonzeros_, and copy the nonzero structure of symT_A into airn_ and ajcn_

6.186.3.11 `void Ipopt::TSymLinearSolver::GiveMatrixToSolver (bool new_matrix, const SymMatrix & sym_A) [private]`

Copy the elements of the matrix in the required format into the array that is provided by the solver interface.

6.186.4 Member Data Documentation

6.186.4.1 `TaggedObject::Tag Ipopt::TSymLinearSolver::atag_ [private]`

Tag for the incoming matrix.

Definition at line 131 of file IpTSymLinearSolver.hpp.

6.186.4.2 `Index Ipopt::TSymLinearSolver::dim_ [private]`

Number of rows and columns of the matrix.

Definition at line 134 of file IpTSymLinearSolver.hpp.

6.186.4.3 Index Ipopt::TSymLinearSolver::nonzeros_triplet_ [private]

Number of nonzeros of the matrix in triplet format.

Note that some elements might appear multiple times in which case the values are added.

Definition at line 139 of file IpTSymLinearSolver.hpp.

6.186.4.4 Index Ipopt::TSymLinearSolver::nonzeros_compressed_ [private]

Number of nonzeros in compressed format.

This is only computed if the sparse linear solver works with the CSR format.

Definition at line 143 of file IpTSymLinearSolver.hpp.

6.186.4.5 bool Ipopt::TSymLinearSolver::have_structure_ [private]

Flag indicating if the internal structures are initialized.

For initialization, this object needs to have seen a matrix

Definition at line 150 of file IpTSymLinearSolver.hpp.

6.186.4.6 bool Ipopt::TSymLinearSolver::linear_scaling_on_demand_ [private]

Flag indicating whether the scaling objected is to be switched on when increased quality is requested.

Definition at line 153 of file IpTSymLinearSolver.hpp.

6.186.4.7 bool Ipopt::TSymLinearSolver::initialized_ [private]

Flag indicating if the InitializeStructure method has been called for the linear solver.

Definition at line 156 of file IpTSymLinearSolver.hpp.

6.186.4.8 SmartPtr<SparseSymLinearSolverInterface> Ipopt::TSymLinearSolver::solver_interface_ [private]

Strategy Object for an interface to a linear solver.

Definition at line 160 of file IpTSymLinearSolver.hpp.

6.186.4.9 SmartPtr<TSymScalingMethod> Ipopt::TSymLinearSolver::scaling_method_ [private]

Strategy Object for a method that computes scaling factors for the matrices.

If NULL, no scaling is performed.

Definition at line 165 of file IpTSymLinearSolver.hpp.

6.186.4.10 double* Ipopt::TSymLinearSolver::scaling_factors_ [private]

Array storing the scaling factors.

Definition at line 167 of file IpTSymLinearSolver.hpp.

6.186.4.11 bool Ipopt::TSymLinearSolver::use_scaling_ [private]

Flag indicating whether scaling should be performed.

Definition at line 169 of file IpTSymLinearSolver.hpp.

6.186.4.12 `bool Ipopt::TSymLinearSolver::just_switched_on_scaling_ [private]`

Flag indicating whether we just switched on the scaling.

Definition at line 171 of file `IpTSymLinearSolver.hpp`.

6.186.4.13 `Index* Ipopt::TSymLinearSolver::airn_ [private]`

row indices of matrix in triplet (MA27) format.

Definition at line 178 of file `IpTSymLinearSolver.hpp`.

6.186.4.14 `Index* Ipopt::TSymLinearSolver::ajcn_ [private]`

column indices of matrix in triplet (MA27) format.

Definition at line 181 of file `IpTSymLinearSolver.hpp`.

6.186.4.15 `SmartPointer<TripletToCSRConverter> Ipopt::TSymLinearSolver::triplet_to_csr_converter_ [private]`

Pointer to object for conversion from triplet to compressed format.

This is only required if the linear solver works with the compressed representation.

Definition at line 185 of file `IpTSymLinearSolver.hpp`.

6.186.4.16 `SparseSymLinearSolverInterface::EMatrixFormat Ipopt::TSymLinearSolver::matrix_format_ [private]`

Flag indicating what matrix data format the solver requires.

Definition at line 187 of file `IpTSymLinearSolver.hpp`.

6.186.4.17 `bool Ipopt::TSymLinearSolver::warm_start_same_structure_ [private]`

Flag indicating whether the [TNLP](#) with identical structure has already been solved before.

Definition at line 194 of file `IpTSymLinearSolver.hpp`.

The documentation for this class was generated from the following file:

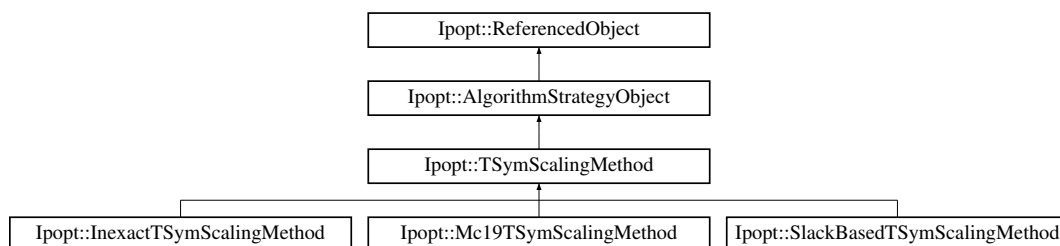
- [Algorithm/LinearSolvers/IpTSymLinearSolver.hpp](#)

6.187 Ipopt::TSymScalingMethod Class Reference

Base class for the method for computing scaling factors for symmetric matrices in triplet format.

```
#include <IpTSymScalingMethod.hpp>
```

Inheritance diagram for `Ipopt::TSymScalingMethod`:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)=0
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [ComputeSymTScalingFactors](#) ([Index](#) n, [Index](#) nnz, const [Index](#) *airn, const [Index](#) *ajcn, const double *a, double *scaling_factors)=0
Method for computing the symmetric scaling factors, given the symmetric matrix in triplet (MA27) format.

Constructor/Destructor

- [TSymScalingMethod](#) ()
- [~TSymScalingMethod](#) ()

Private Member Functions

Default Compiler Generated Methods (Hidden to avoid

implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [TSymScalingMethod](#) (const [TSymScalingMethod](#) &)
Copy Constructor.
- void [operator=](#) (const [TSymScalingMethod](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.187.1 Detailed Description

Base class for the method for computing scaling factors for symmetric matrices in triplet format.

Definition at line 23 of file `IpTSymScalingMethod.hpp`.

6.187.2 Constructor & Destructor Documentation

6.187.2.1 `Ipopt::TSymScalingMethod::TSymScalingMethod () [inline]`

Definition at line 28 of file `IpTSymScalingMethod.hpp`.

6.187.2.2 `Ipopt::TSymScalingMethod::~~TSymScalingMethod () [inline]`

Definition at line 31 of file `IpTSymScalingMethod.hpp`.

6.187.2.3 `Ipopt::TSymScalingMethod::TSymScalingMethod (const TSymScalingMethod &) [private]`

Copy Constructor.

6.187.3 Member Function Documentation

6.187.3.1 `virtual bool Ipopt::TSymScalingMethod::InitializeImpl (const OptionsList & options, const std::string & prefix) [pure virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::AlgorithmStrategyObject](#).

Implemented in [Ipopt::InexactTSymScalingMethod](#), [Ipopt::SlackBasedTSymScalingMethod](#), and [Ipopt::Mc19TSymScalingMethod](#).

6.187.3.2 `virtual bool Ipopt::TSymScalingMethod::ComputeSymTScalingFactors (Index n, Index nnz, const Index * ainr, const Index * ajcn, const double * a, double * scaling_factors)` `[pure virtual]`

Method for computing the symmetric scaling factors, given the symmetric matrix in triplet (MA27) format.

6.187.3.3 `void Ipopt::TSymScalingMethod::operator= (const TSymScalingMethod &)` `[private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

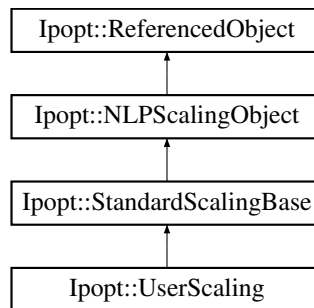
- [Algorithm/LinearSolvers/IpTSymScalingMethod.hpp](#)

6.188 Ipopt::UserScaling Class Reference

This class does problem scaling by getting scaling parameters from the user (through the [NLP](#) interface).

```
#include <IpUserScaling.hpp>
```

Inheritance diagram for Ipopt::UserScaling:



Public Member Functions

Constructors/Destructors

- [UserScaling](#) (const [SmartPtr](#)< const [NLP](#) > &nlp)
- virtual [~UserScaling](#) ()
Default destructor.

Protected Member Functions

- virtual void [DetermineScalingParametersImpl](#) (const [SmartPtr](#)< const [VectorSpace](#) > x_space, const [SmartPtr](#)< const [VectorSpace](#) > c_space, const [SmartPtr](#)< const [VectorSpace](#) > d_space, const [SmartPtr](#)< const [MatrixSpace](#) > jac_c_space, const [SmartPtr](#)< const [MatrixSpace](#) > jac_d_space, const [SmartPtr](#)< const [SymMatrixSpace](#) > h_space, const [Matrix](#) &Px_L, const [Vector](#) &x_L, const [Matrix](#) &Px_U, const [Vector](#) &x_U, [Number](#) &df, [SmartPtr](#)< [Vector](#) > &dx, [SmartPtr](#)< [Vector](#) > &dc, [SmartPtr](#)< [Vector](#) > &dd)

*This is the method that has to be overloaded by a particular scaling method that somehow computes the scaling vectors *dx*, *dc*, and *dd*.*

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- `UserScaling` (const `UserScaling` &)
Copy Constructor.
- `void operator=` (const `UserScaling` &)
Overloaded Equals Operator.

Private Attributes

- `SmartPointer< const NLP > nlp_`
pointer to the `NLP` to get scaling parameters

Additional Inherited Members

6.188.1 Detailed Description

This class does problem scaling by getting scaling parameters from the user (through the `NLP` interface).

Definition at line 20 of file `IpUserScaling.hpp`.

6.188.2 Constructor & Destructor Documentation

6.188.2.1 `Ipopt::UserScaling::UserScaling (const SmartPtr< const NLP > & nlp) [inline]`

Definition at line 25 of file `IpUserScaling.hpp`.

6.188.2.2 `virtual Ipopt::UserScaling::~UserScaling () [inline],[virtual]`

Default destructor.

Definition at line 32 of file `IpUserScaling.hpp`.

6.188.2.3 `Ipopt::UserScaling::UserScaling (const UserScaling &) [private]`

Copy Constructor.

6.188.3 Member Function Documentation

6.188.3.1 `virtual void Ipopt::UserScaling::DetermineScalingParametersImpl (const SmartPtr< const VectorSpace > x_space, const SmartPtr< const VectorSpace > c_space, const SmartPtr< const VectorSpace > d_space, const SmartPtr< const MatrixSpace > jac_c_space, const SmartPtr< const MatrixSpace > jac_d_space, const SmartPtr< const SymMatrixSpace > h_space, const Matrix & Px_L, const Vector & x_L, const Matrix & Px_U, const Vector & x_U, Number & df, SmartPtr< Vector > & dx, SmartPtr< Vector > & dc, SmartPtr< Vector > & dd) [protected],[virtual]`

This is the method that has to be overloaded by a particular scaling method that somehow computes the scaling vectors `dx`, `dc`, and `dd`.

The pointers to those vectors can be NULL, in which case no scaling for that item will be done later.

Implements [Ipopt::StandardScalingBase](#).

6.188.3.2 void Ipopt::UserScaling::operator= (const UserScaling &) [private]

Overloaded Equals Operator.

6.188.4 Member Data Documentation

6.188.4.1 SmartPtr<const NLP> Ipopt::UserScaling::nlp_ [private]

pointer to the [NLP](#) to get scaling parameters

Definition at line 70 of file IpUserScaling.hpp.

The documentation for this class was generated from the following file:

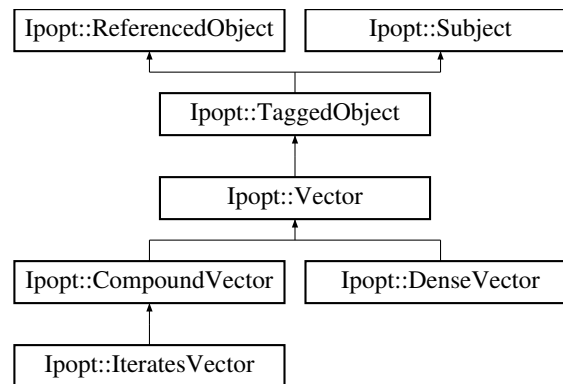
- [Algorithm/IpUserScaling.hpp](#)

6.189 Ipopt::Vector Class Reference

[Vector](#) Base Class.

```
#include <IpVector.hpp>
```

Inheritance diagram for Ipopt::Vector:



Public Member Functions

- [Vector](#) * [MakeNew](#) () const
Create new [Vector](#) of the same type with uninitialized data.
- [Vector](#) * [MakeNewCopy](#) () const
Create new [Vector](#) of the same type and copy the data over.
- bool [IsValidNumbers](#) () const
Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Constructor/Destructor

- [Vector](#) (const [VectorSpace](#) *owner_space)

- *Constructor.*
- virtual `~Vector()`
- *Destructor.*

Standard BLAS-1 Operations

(derived classes do NOT overload these methods, instead, overload the protected versions of these methods).

- void `Copy` (const `Vector` &x)
Copy the data of the vector x into this vector (DCOPY).
- void `Scal` (`Number` alpha)
Scales the vector by scalar alpha (DSCAL)
- void `Axpy` (`Number` alpha, const `Vector` &x)
Add the multiple alpha of vector x to this vector (DAXPY)
- `Number Dot` (const `Vector` &x) const
Computes inner product of vector x with this (DDOT)
- `Number Nrm2` () const
Computes the 2-norm of this vector (DNRM2)
- `Number Asum` () const
Computes the 1-norm of this vector (DASUM)
- `Number Amax` () const
Computes the max-norm of this vector (based on IDAMAX)

Additional (Non-BLAS) Vector Methods

(derived classes do NOT overload these methods, instead, overload the protected versions of these methods).

- void `Set` (`Number` alpha)
Set each element in the vector to the scalar alpha.
- void `ElementWiseDivide` (const `Vector` &x)
Element-wise division $y_i \leftarrow y_i / x_i$.
- void `ElementWiseMultiply` (const `Vector` &x)
*Element-wise multiplication $y_i \leftarrow y_i * x_i$.*
- void `ElementWiseMax` (const `Vector` &x)
Element-wise max against entries in x.
- void `ElementWiseMin` (const `Vector` &x)
Element-wise min against entries in x.
- void `ElementWiseReciprocal` ()
Reciprocates the entries in the vector.
- void `ElementWiseAbs` ()
Absolute values of the entries in the vector.
- void `ElementWiseSqrt` ()
Element-wise square root of the entries in the vector.
- void `ElementWiseSgn` ()
*Replaces the vector values with their sgn values
(-1 if $x_i < 0$, 0 if $x_i == 0$, and 1 if $x_i > 0$)*
- void `AddScalar` (`Number` scalar)
Add scalar to every vector component.
- `Number Max` () const
Returns the maximum value in the vector.
- `Number Min` () const
Returns the minimum value in the vector.
- `Number Sum` () const
Returns the sum of the vector entries.
- `Number SumLogs` () const
Returns the sum of the logs of each vector entry.

Methods for specialized operations. A prototype

implementation is provided, but for efficient implementation those should be specially implemented.

- void `AddOneVector` (`Number` a, const `Vector` &v1, `Number` c)
*Add one vector, $y = a * v1 + c * y$.*
- void `AddTwoVectors` (`Number` a, const `Vector` &v1, `Number` b, const `Vector` &v2, `Number` c)
*Add two vectors, $y = a * v1 + b * v2 + c * y$.*
- `Number` `FracToBound` (const `Vector` &delta, `Number` tau) const
Fraction to the boundary parameter.
- void `AddVectorQuotient` (`Number` a, const `Vector` &z, const `Vector` &s, `Number` c)
*Add the quotient of two vectors, $y = a * z/s + c * y$.*

Accessor methods

- `Index` `Dim` () const
Dimension of the `Vector`.
- `SmartPtr`< const `VectorSpace` > `OwnerSpace` () const
Return the owner `VectorSpace`.

Output methods

(derived classes do NOT overload these methods, instead, overload the protected versions of these methods).

- void `Print` (`SmartPtr`< const `Journalist` > jnlst, `EJournalLevel` level, `EJournalCategory` category, const std::string &name, `Index` indent=0, const std::string &prefix="") const
Print the entire vector.
- void `Print` (const `Journalist` &jnlst, `EJournalLevel` level, `EJournalCategory` category, const std::string &name, `Index` indent=0, const std::string &prefix="") const

Protected Member Functions**implementation methods (derived classes MUST**

overload these pure virtual protected methods.)

- virtual void `CopyImpl` (const `Vector` &x)=0
Copy the data of the vector x into this vector (DCOPY).
- virtual void `ScallImpl` (`Number` alpha)=0
Scales the vector by scalar alpha (DSCAL)
- virtual void `AxpyImpl` (`Number` alpha, const `Vector` &x)=0
Add the multiple alpha of vector x to this vector (DAXPY)
- virtual `Number` `DotImpl` (const `Vector` &x) const =0
Computes inner product of vector x with this (DDOT)
- virtual `Number` `Nrm2Impl` () const =0
Computes the 2-norm of this vector (DNRM2)
- virtual `Number` `AsumImpl` () const =0
Computes the 1-norm of this vector (DASUM)
- virtual `Number` `AmaxImpl` () const =0
Computes the max-norm of this vector (based on IDAMAX)
- virtual void `SetImpl` (`Number` alpha)=0
Set each element in the vector to the scalar alpha.
- virtual void `ElementWiseDivideImpl` (const `Vector` &x)=0
Element-wise division $y_i \leftarrow y_i / x_i$.
- virtual void `ElementWiseMultiplyImpl` (const `Vector` &x)=0
*Element-wise multiplication $y_i \leftarrow y_i * x_i$.*
- virtual void `ElementWiseMaxImpl` (const `Vector` &x)=0

- *Element-wise max against entries in x.*
virtual void [ElementWiseMinImpl](#) (const [Vector](#) &x)=0
- *Element-wise min against entries in x.*
virtual void [ElementWiseReciprocalImpl](#) ()=0
- *Reciprocates the elements of the vector.*
virtual void [ElementWiseAbsImpl](#) ()=0
- *Take elementwise absolute values of the elements of the vector.*
virtual void [ElementWiseSqrtImpl](#) ()=0
- *Take elementwise square-root of the elements of the vector.*
virtual void [ElementWiseSgnImpl](#) ()=0
- *Replaces entries with sgn of the entry.*
virtual void [AddScalarImpl](#) ([Number](#) scalar)=0
- *Add scalar to every component of vector.*
virtual [Number](#) [MaxImpl](#) () const =0
- *Max value in the vector.*
virtual [Number](#) [MinImpl](#) () const =0
- *Min number in the vector.*
virtual [Number](#) [SumImpl](#) () const =0
- *Sum of entries in the vector.*
virtual [Number](#) [SumLogsImpl](#) () const =0
- *Sum of logs of entries in the vector.*
virtual void [AddTwoVectorsImpl](#) ([Number](#) a, const [Vector](#) &v1, [Number](#) b, const [Vector](#) &v2, [Number](#) c)
- *Add two vectors ($a * v1 + b * v2$).*
virtual [Number](#) [FracToBoundImpl](#) (const [Vector](#) &delta, [Number](#) tau) const
- *Fraction to boundary parameter.*
virtual void [AddVectorQuotientImpl](#) ([Number](#) a, const [Vector](#) &z, const [Vector](#) &s, [Number](#) c)
- *Add the quotient of two vectors.*
virtual bool [IsValidNumbersImpl](#) () const
- *Method for determining if all stored numbers are valid (i.e., no Inf or Nan).*
virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const =0
- *Print the entire vector.*

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [Vector](#) ()
Default constructor.
- [Vector](#) (const [Vector](#) &)
Copy constructor.
- [Vector](#) & [operator=](#) (const [Vector](#) &)
Overloaded Equals Operator.

Private Attributes

- const [SmartPtr](#)< const [VectorSpace](#) > [owner_space_](#)
[Vector](#) Space.

CachedResults data members

- [CachedResults< Number > dot_cache_](#)
Cache for dot products.
- [TaggedObject::Tag nrm2_cache_tag_](#)
- [Number cached_nrm2_](#)
- [TaggedObject::Tag asum_cache_tag_](#)
- [Number cached_asum_](#)
- [TaggedObject::Tag amax_cache_tag_](#)
- [Number cached_amax_](#)
- [TaggedObject::Tag max_cache_tag_](#)
- [Number cached_max_](#)
- [TaggedObject::Tag min_cache_tag_](#)
- [Number cached_min_](#)
- [TaggedObject::Tag sum_cache_tag_](#)
- [Number cached_sum_](#)
- [TaggedObject::Tag sumlogs_cache_tag_](#)
- [Number cached_sumlogs_](#)
- [TaggedObject::Tag valid_cache_tag_](#)
- [bool cached_valid_](#)

Additional Inherited Members

6.189.1 Detailed Description

[Vector](#) Base Class.

This is the base class for all derived vector types. Those vectors are meant to store entities like iterates, Lagrangian multipliers, constraint values etc. The implementation of a vector type depends on the computational environment (e.g. just a double array on a shared memory machine, or distributed double arrays for a distributed memory machine.)

Deriving from [Vector](#): This class inherits from tagged object to implement an advanced caching scheme. Because of this, the [TaggedObject](#) method [ObjectChanged\(\)](#) must be called each time the [Vector](#) changes. If you overload the XX-XX_Impl protected methods, this taken care of (along with caching if possible) for you. If you have additional methods in your derived class that change the underlying data (vector values), you MUST remember to call [ObjectChanged\(\)](#) AFTER making the change!

Definition at line 47 of file IpVector.hpp.

6.189.2 Constructor & Destructor Documentation

6.189.2.1 `Ipopt::Vector::Vector (const VectorSpace * owner_space) [inline]`

Constructor.

It has to be given a pointer to the corresponding [VectorSpace](#).

Definition at line 445 of file IpVector.hpp.

6.189.2.2 `Ipopt::Vector::~Vector () [inline],[virtual]`

Destructor.

Definition at line 441 of file IpVector.hpp.

6.189.2.3 `Ipopt::Vector::Vector () [private]`

Default constructor.

6.189.2.4 Ipopt::Vector::Vector (const Vector &) [private]

Copy constructor.

6.189.3 Member Function Documentation

6.189.3.1 Vector * Ipopt::Vector::MakeNew () const [inline]

Create new [Vector](#) of the same type with uninitialized data.

Definition at line 456 of file IpVector.hpp.

6.189.3.2 Vector * Ipopt::Vector::MakeNewCopy () const [inline]

Create new [Vector](#) of the same type and copy the data over.

Definition at line 462 of file IpVector.hpp.

6.189.3.3 void Ipopt::Vector::Copy (const Vector & x) [inline]

Copy the data of the vector x into this vector (DCOPY).

Definition at line 471 of file IpVector.hpp.

6.189.3.4 void Ipopt::Vector::Scal (Number alpha)

Scales the vector by scalar alpha (DSCAL)

6.189.3.5 void Ipopt::Vector::Axy (Number alpha, const Vector & x) [inline]

Add the multiple alpha of vector x to this vector (DAXPY)

Definition at line 509 of file IpVector.hpp.

6.189.3.6 Number Ipopt::Vector::Dot (const Vector & x) const [inline]

Computes inner product of vector x with this (DDOT)

Definition at line 516 of file IpVector.hpp.

6.189.3.7 Number Ipopt::Vector::Nrm2 () const [inline]

Computes the 2-norm of this vector (DNRM2)

Definition at line 535 of file IpVector.hpp.

6.189.3.8 Number Ipopt::Vector::Asum () const [inline]

Computes the 1-norm of this vector (DASUM)

Definition at line 545 of file IpVector.hpp.

6.189.3.9 Number Ipopt::Vector::Amax () const [inline]

Computes the max-norm of this vector (based on IDAMAX)

Definition at line 555 of file IpVector.hpp.

6.189.3.10 void Ipopt::Vector::Set (**Number** *alpha*) [inline]

Set each element in the vector to the scalar alpha.

Definition at line 592 of file IpVector.hpp.

6.189.3.11 void Ipopt::Vector::ElementWiseDivide (const **Vector** & *x*) [inline]

Element-wise division $y_i \leftarrow y_i / x_i$.

Definition at line 600 of file IpVector.hpp.

6.189.3.12 void Ipopt::Vector::ElementWiseMultiply (const **Vector** & *x*) [inline]

Element-wise multiplication $y_i \leftarrow y_i * x_i$.

Definition at line 607 of file IpVector.hpp.

6.189.3.13 void Ipopt::Vector::ElementWiseMax (const **Vector** & *x*) [inline]

Element-wise max against entries in x.

Definition at line 621 of file IpVector.hpp.

6.189.3.14 void Ipopt::Vector::ElementWiseMin (const **Vector** & *x*) [inline]

Element-wise min against entries in x.

Definition at line 629 of file IpVector.hpp.

6.189.3.15 void Ipopt::Vector::ElementWiseReciprocal () [inline]

Reciprocates the entries in the vector.

Definition at line 614 of file IpVector.hpp.

6.189.3.16 void Ipopt::Vector::ElementWiseAbs () [inline]

Absolute values of the entries in the vector.

Definition at line 637 of file IpVector.hpp.

6.189.3.17 void Ipopt::Vector::ElementWiseSqrt () [inline]

Element-wise square root of the entries in the vector.

Definition at line 645 of file IpVector.hpp.

6.189.3.18 void Ipopt::Vector::ElementWiseSgn () [inline]

Replaces the vector values with their sgn values

(-1 if $x_i < 0$, 0 if $x_i == 0$, and 1 if $x_i > 0$)

Definition at line 585 of file IpVector.hpp.

6.189.3.19 void Ipopt::Vector::AddScalar (**Number** *scalar*) [inline]

Add scalar to every vector component.

Definition at line 652 of file IpVector.hpp.

6.189.3.20 `Number lpopt::Vector::Max () const [inline]`

Returns the maximum value in the vector.

Definition at line 660 of file lpVector.hpp.

6.189.3.21 `Number lpopt::Vector::Min () const [inline]`

Returns the minimum value in the vector.

Definition at line 670 of file lpVector.hpp.

6.189.3.22 `Number lpopt::Vector::Sum () const [inline]`

Returns the sum of the vector entries.

Definition at line 565 of file lpVector.hpp.

6.189.3.23 `Number lpopt::Vector::SumLogs () const [inline]`

Returns the sum of the logs of each vector entry.

Definition at line 575 of file lpVector.hpp.

6.189.3.24 `void lpopt::Vector::AddOneVector (Number a, const Vector & v1, Number c) [inline]`

Add one vector, $y = a * v1 + c * y$.

This is automatically reduced to call AddTwoVectors.

Definition at line 680 of file lpVector.hpp.

6.189.3.25 `void lpopt::Vector::AddTwoVectors (Number a, const Vector & v1, Number b, const Vector & v2, Number c) [inline]`

Add two vectors, $y = a * v1 + b * v2 + c * y$.

Here, this vector is y

Definition at line 686 of file lpVector.hpp.

6.189.3.26 `Number lpopt::Vector::FracToBound (const Vector & delta, Number tau) const [inline]`

Fraction to the boundary parameter.

Computes $\alpha = \max\{\bar{\alpha} \in (0, 1] : x + \bar{\alpha}\Delta \geq (1 - \tau)x\}$

Definition at line 694 of file lpVector.hpp.

6.189.3.27 `void lpopt::Vector::AddVectorQuotient (Number a, const Vector & z, const Vector & s, Number c) [inline]`

Add the quotient of two vectors, $y = a * z/s + c * y$.

Definition at line 714 of file lpVector.hpp.

6.189.3.28 `bool lpopt::Vector::IsValidNumbers () const [inline]`

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

Definition at line 722 of file lpVector.hpp.

6.189.3.29 **Index** `Ipopt::Vector::Dim () const` `[inline]`

Dimension of the [Vector](#).

Definition at line 732 of file `IpVector.hpp`.

6.189.3.30 **SmartPtr**< **const VectorSpace** > `Ipopt::Vector::OwnerSpace () const` `[inline]`

Return the owner [VectorSpace](#).

Definition at line 738 of file `IpVector.hpp`.

6.189.3.31 **void** `Ipopt::Vector::Print (SmartPtr< const Journalist > jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent = 0, const std::string & prefix = " ") const`

Print the entire vector.

6.189.3.32 **void** `Ipopt::Vector::Print (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent = 0, const std::string & prefix = " ") const`

6.189.3.33 **virtual void** `Ipopt::Vector::CopyImpl (const Vector & x)` `[protected]`, `[pure virtual]`

Copy the data of the vector *x* into this vector (DCOPY).

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.34 **virtual void** `Ipopt::Vector::ScalImpl (Number alpha)` `[protected]`, `[pure virtual]`

Scales the vector by scalar *alpha* (DSCAL)

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.35 **virtual void** `Ipopt::Vector::Axpympl (Number alpha, const Vector & x)` `[protected]`, `[pure virtual]`

Add the multiple *alpha* of vector *x* to this vector (DAXPY)

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.36 **virtual Number** `Ipopt::Vector::DotImpl (const Vector & x) const` `[protected]`, `[pure virtual]`

Computes inner product of vector *x* with this (DDOT)

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.37 **virtual Number** `Ipopt::Vector::Nrm2Impl () const` `[protected]`, `[pure virtual]`

Computes the 2-norm of this vector (DNRM2)

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.38 **virtual Number** `Ipopt::Vector::AsumImpl () const` `[protected]`, `[pure virtual]`

Computes the 1-norm of this vector (DASUM)

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.39 **virtual Number** `Ipopt::Vector::AmaxImpl () const` `[protected]`, `[pure virtual]`

Computes the max-norm of this vector (based on IDAMAX)

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.40 `virtual void lpopt::Vector::SetImpl (Number alpha)` [protected],[pure virtual]

Set each element in the vector to the scalar alpha.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.41 `virtual void lpopt::Vector::ElementWiseDivideImpl (const Vector & x)` [protected],[pure virtual]

Element-wise division $y_i \leftarrow y_i / x_i$.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.42 `virtual void lpopt::Vector::ElementWiseMultiplyImpl (const Vector & x)` [protected],[pure virtual]

Element-wise multiplication $y_i \leftarrow y_i * x_i$.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.43 `virtual void lpopt::Vector::ElementWiseMaxImpl (const Vector & x)` [protected],[pure virtual]

Element-wise max against entries in x.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.44 `virtual void lpopt::Vector::ElementWiseMinImpl (const Vector & x)` [protected],[pure virtual]

Element-wise min against entries in x.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.45 `virtual void lpopt::Vector::ElementWiseReciprocalImpl ()` [protected],[pure virtual]

Reciprocates the elements of the vector.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.46 `virtual void lpopt::Vector::ElementWiseAbsImpl ()` [protected],[pure virtual]

Take elementwise absolute values of the elements of the vector.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.47 `virtual void lpopt::Vector::ElementWiseSqrtImpl ()` [protected],[pure virtual]

Take elementwise square-root of the elements of the vector.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.48 `virtual void lpopt::Vector::ElementWiseSgnImpl ()` [protected],[pure virtual]

Replaces entries with sgn of the entry.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.49 `virtual void lpopt::Vector::AddScalarImpl (Number scalar)` [protected],[pure virtual]

Add scalar to every component of vector.

Implemented in [lpopt::DenseVector](#), and [lpopt::CompoundVector](#).

6.189.3.50 `virtual Number Ipopt::Vector::MaxImpl () const` [protected], [pure virtual]

Max value in the vector.

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.51 `virtual Number Ipopt::Vector::MinImpl () const` [protected], [pure virtual]

Min number in the vector.

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.52 `virtual Number Ipopt::Vector::SumImpl () const` [protected], [pure virtual]

Sum of entries in the vector.

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.53 `virtual Number Ipopt::Vector::SumLogsImpl () const` [protected], [pure virtual]

Sum of logs of entries in the vector.

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.54 `virtual void Ipopt::Vector::AddTwoVectorsImpl (Number a, const Vector & v1, Number b, const Vector & v2, Number c)` [protected], [virtual]

Add two vectors ($a * v1 + b * v2$).

Result is stored in this vector.

Reimplemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.55 `virtual Number Ipopt::Vector::FracToBoundImpl (const Vector & delta, Number tau) const` [protected], [virtual]

Fraction to boundary parameter.

Reimplemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.56 `virtual void Ipopt::Vector::AddVectorQuotientImpl (Number a, const Vector & z, const Vector & s, Number c)` [protected], [virtual]

Add the quotient of two vectors.

Reimplemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.57 `virtual bool Ipopt::Vector::IsValidNumbersImpl () const` [protected], [virtual]

Method for determining if all stored numbers are valid (i.e., no Inf or Nan).

A default implementation using Asum is provided.

Reimplemented in [Ipopt::CompoundVector](#).

6.189.3.58 `virtual void Ipopt::Vector::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const` [protected], [pure virtual]

Print the entire vector.

Implemented in [Ipopt::DenseVector](#), and [Ipopt::CompoundVector](#).

6.189.3.59 **Vector& lpopt::Vector::operator= (const Vector &)** [private]

Overloaded Equals Operator.

6.189.4 Member Data Documentation

6.189.4.1 **const SmartPtr<const VectorSpace> lpopt::Vector::owner_space_** [private]

[Vector](#) Space.

Definition at line 343 of file IpVector.hpp.

6.189.4.2 **CachedResults<Number> lpopt::Vector::dot_cache_** [mutable],[private]

Cache for dot products.

Definition at line 348 of file IpVector.hpp.

6.189.4.3 **TaggedObject::Tag lpopt::Vector::nrm2_cache_tag_** [mutable],[private]

Definition at line 350 of file IpVector.hpp.

6.189.4.4 **Number lpopt::Vector::cached_nrm2_** [mutable],[private]

Definition at line 351 of file IpVector.hpp.

6.189.4.5 **TaggedObject::Tag lpopt::Vector::asum_cache_tag_** [mutable],[private]

Definition at line 353 of file IpVector.hpp.

6.189.4.6 **Number lpopt::Vector::cached_asum_** [mutable],[private]

Definition at line 354 of file IpVector.hpp.

6.189.4.7 **TaggedObject::Tag lpopt::Vector::amax_cache_tag_** [mutable],[private]

Definition at line 356 of file IpVector.hpp.

6.189.4.8 **Number lpopt::Vector::cached_amax_** [mutable],[private]

Definition at line 357 of file IpVector.hpp.

6.189.4.9 **TaggedObject::Tag lpopt::Vector::max_cache_tag_** [mutable],[private]

Definition at line 359 of file IpVector.hpp.

6.189.4.10 **Number lpopt::Vector::cached_max_** [mutable],[private]

Definition at line 360 of file IpVector.hpp.

6.189.4.11 **TaggedObject::Tag lpopt::Vector::min_cache_tag_** [mutable],[private]

Definition at line 362 of file IpVector.hpp.

6.189.4.12 **Number lpopt::Vector::cached_min_** [mutable],[private]

Definition at line 363 of file IpVector.hpp.

6.189.4.13 **TaggedObject::Tag** Ipopt::Vector::sum_cache_tag_ [mutable], [private]

Definition at line 365 of file IpVector.hpp.

6.189.4.14 **Number** Ipopt::Vector::cached_sum_ [mutable], [private]

Definition at line 366 of file IpVector.hpp.

6.189.4.15 **TaggedObject::Tag** Ipopt::Vector::sumlogs_cache_tag_ [mutable], [private]

Definition at line 368 of file IpVector.hpp.

6.189.4.16 **Number** Ipopt::Vector::cached_sumlogs_ [mutable], [private]

Definition at line 369 of file IpVector.hpp.

6.189.4.17 **TaggedObject::Tag** Ipopt::Vector::valid_cache_tag_ [mutable], [private]

Definition at line 371 of file IpVector.hpp.

6.189.4.18 **bool** Ipopt::Vector::cached_valid_ [mutable], [private]

Definition at line 372 of file IpVector.hpp.

The documentation for this class was generated from the following file:

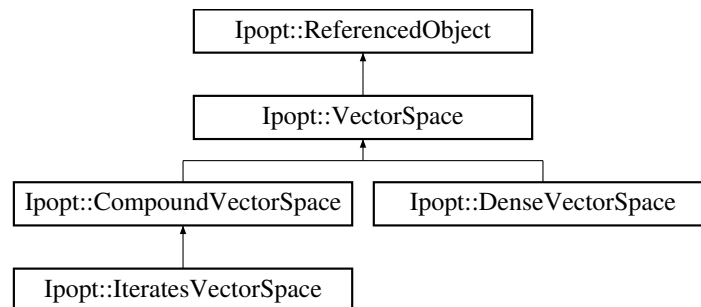
- [LinAlg/IpVector.hpp](#)

6.190 Ipopt::VectorSpace Class Reference

[VectorSpace](#) base class, corresponding to the [Vector](#) base class.

```
#include <IpVector.hpp>
```

Inheritance diagram for Ipopt::VectorSpace:



Public Member Functions

- virtual [Vector](#) * [MakeNew](#) () const =0
Pure virtual method for creating a new [Vector](#) of the corresponding type.
- [Index Dim](#) () const
Accessor function for the dimension of the vectors of this type.

Constructors/Destructors

- [VectorSpace](#) ([Index](#) dim)
Constructor, given the dimension of all vectors generated by this [VectorSpace](#).
- virtual [~VectorSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [VectorSpace](#) ()
default constructor
- [VectorSpace](#) (const [VectorSpace](#) &)
Copy constructor.
- [VectorSpace](#) & [operator=](#) (const [VectorSpace](#) &)
Overloaded Equals Operator.

Private Attributes

- const [Index](#) dim_
Dimension of the vectors in this vector space.

6.190.1 Detailed Description

[VectorSpace](#) base class, corresponding to the [Vector](#) base class.

For each [Vector](#) implementation, a corresponding [VectorSpace](#) has to be implemented. A [VectorSpace](#) is able to create new Vectors of a specific type. The [VectorSpace](#) should also store information that is common to all Vectors of that type. For example, the dimension of a [Vector](#) is stored in the [VectorSpace](#) base class.

Definition at line 390 of file `IpVector.hpp`.

6.190.2 Constructor & Destructor Documentation

6.190.2.1 `Ipopt::VectorSpace::VectorSpace (Index dim) [inline]`

Constructor, given the dimension of all vectors generated by this [VectorSpace](#).

Definition at line 744 of file `IpVector.hpp`.

6.190.2.2 `virtual Ipopt::VectorSpace::~~VectorSpace () [inline],[virtual]`

Destructor.

Definition at line 401 of file `IpVector.hpp`.

6.190.2.3 `Ipopt::VectorSpace::VectorSpace () [private]`

default constructor

6.190.2.4 Ipopt::VectorSpace::VectorSpace (const VectorSpace &) [private]

Copy constructor.

6.190.3 Member Function Documentation

6.190.3.1 virtual Vector* Ipopt::VectorSpace::MakeNew () const [pure virtual]

Pure virtual method for creating a new [Vector](#) of the corresponding type.

Implemented in [Ipopt::IteratesVectorSpace](#), [Ipopt::DenseVectorSpace](#), and [Ipopt::CompoundVectorSpace](#).

6.190.3.2 Index Ipopt::VectorSpace::Dim () const [inline]

Accessor function for the dimension of the vectors of this type.

Definition at line 411 of file IpVector.hpp.

6.190.3.3 VectorSpace& Ipopt::VectorSpace::operator= (const VectorSpace &) [private]

Overloaded Equals Operator.

6.190.4 Member Data Documentation

6.190.4.1 const Index Ipopt::VectorSpace::dim_ [private]

Dimension of the vectors in this vector space.

Definition at line 436 of file IpVector.hpp.

The documentation for this class was generated from the following file:

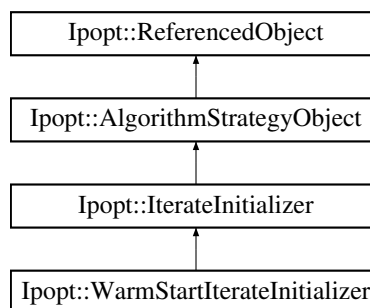
- [LinAlg/IpVector.hpp](#)

6.191 Ipopt::WarmStartIterateInitializer Class Reference

Class implementing an initialization procedure for warm starts.

```
#include <IpWarmStartIterateInitializer.hpp>
```

Inheritance diagram for Ipopt::WarmStartIterateInitializer:



Public Member Functions

- virtual bool [InitializeImpl](#) (const [OptionsList](#) &options, const std::string &prefix)
overloaded from [AlgorithmStrategyObject](#)
- virtual bool [SetInitialIterates](#) ()
Compute the initial iterates and set the into the curr field of the ip_data object.

Constructors/Destructors

- [WarmStartIterateInitializer](#) ()
Constructor.
- virtual [~WarmStartIterateInitializer](#) ()
Default destructor.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods used by IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [WarmStartIterateInitializer](#) (const [WarmStartIterateInitializer](#) &)
Copy Constructor.
- void [operator=](#) (const [WarmStartIterateInitializer](#) &)
Overloaded Equals Operator.

Auxilliary functions

- void [process_target_mu](#) ([Number](#) factor, const [Vector](#) &curr_vars, const [Vector](#) &curr_slacks, const [Vector](#) &curr_mults, const [Matrix](#) &P, [SmartPtr](#)< const [Vector](#) > &ret_vars, [SmartPtr](#)< const [Vector](#) > &ret_mults)
- void [adapt_to_target_mu](#) ([Vector](#) &new_s, [Vector](#) &new_z, [Number](#) target_mu)

Private Attributes

Algorithmic Parameters

- [Number warm_start_bound_push_](#)
Abolsute parameters for bumping x0 in warm start mode.
- [Number warm_start_bound_frac_](#)
Relative parameters for bumping x0 in warm start mode.
- [Number warm_start_slack_bound_push_](#)
Abolsute parameters for bumping s0 in warm start mode.
- [Number warm_start_slack_bound_frac_](#)
Relative parameters for bumping s0 in warm start mode.
- [Number warm_start_mult_bound_push_](#)
Parameters for bumping initial bound multipliers.

- [Number warm_start_mult_init_max_](#)
Maximal size of entries in bound and equality constraint multipliers in magnitude.
- [Number warm_start_target_mu_](#)
Target values for the barrier parameter in warm start option.
- [bool warm_start_entire_iterate_](#)
Indicator for which method in the [NLP](#) should be used to get the warm start.

Additional Inherited Members

6.191.1 Detailed Description

Class implementing an initialization procedure for warm starts.

Definition at line 20 of file IpWarmStartIterateInitializer.hpp.

6.191.2 Constructor & Destructor Documentation

6.191.2.1 Ipopt::WarmStartIterateInitializer::WarmStartIterateInitializer ()

Constructor.

6.191.2.2 virtual Ipopt::WarmStartIterateInitializer::~~WarmStartIterateInitializer () [inline], [virtual]

Default destructor.

Definition at line 29 of file IpWarmStartIterateInitializer.hpp.

6.191.2.3 Ipopt::WarmStartIterateInitializer::WarmStartIterateInitializer (const WarmStartIterateInitializer &) [private]

Copy Constructor.

6.191.3 Member Function Documentation

6.191.3.1 virtual bool Ipopt::WarmStartIterateInitializer::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::IterateInitializer](#).

6.191.3.2 virtual bool Ipopt::WarmStartIterateInitializer::SetInitialIterates () [virtual]

Compute the initial iterates and set the into the curr field of the ip_data object.

Implements [Ipopt::IterateInitializer](#).

6.191.3.3 static void Ipopt::WarmStartIterateInitializer::RegisterOptions (SmartPtr< RegisteredOptions > roptions) [static]

Methods used by IpoptType.

6.191.3.4 void Ipopt::WarmStartIterateInitializer::operator= (const WarmStartIterateInitializer &) [private]

Overloaded Equals Operator.

6.191.3.5 `void Ipopt::WarmStartIterateInitializer::process_target_mu (Number factor, const Vector & curr_vars, const Vector & curr_slacks, const Vector & curr_mults, const Matrix & P, SmartPtr< const Vector > & ret_vars, SmartPtr< const Vector > & ret_mults) [private]`

6.191.3.6 `void Ipopt::WarmStartIterateInitializer::adapt_to_target_mu (Vector & new_s, Vector & new_z, Number target_mu) [private]`

6.191.4 Member Data Documentation

6.191.4.1 `Number Ipopt::WarmStartIterateInitializer::warm_start_bound_push_ [private]`

Abolsute parameters for bumping x0 in warm start mode.

Definition at line 64 of file IpWarmStartIterateInitializer.hpp.

6.191.4.2 `Number Ipopt::WarmStartIterateInitializer::warm_start_bound_frac_ [private]`

Relative parameters for bumping x0 in warm start mode.

Definition at line 66 of file IpWarmStartIterateInitializer.hpp.

6.191.4.3 `Number Ipopt::WarmStartIterateInitializer::warm_start_slack_bound_push_ [private]`

Abolsute parameters for bumping s0 in warm start mode.

Definition at line 68 of file IpWarmStartIterateInitializer.hpp.

6.191.4.4 `Number Ipopt::WarmStartIterateInitializer::warm_start_slack_bound_frac_ [private]`

Relative parameters for bumping s0 in warm start mode.

Definition at line 70 of file IpWarmStartIterateInitializer.hpp.

6.191.4.5 `Number Ipopt::WarmStartIterateInitializer::warm_start_mult_bound_push_ [private]`

Parameters for bumping initial bound multipliers.

Definition at line 72 of file IpWarmStartIterateInitializer.hpp.

6.191.4.6 `Number Ipopt::WarmStartIterateInitializer::warm_start_mult_init_max_ [private]`

Maximal size of entries in bound and equality constraint multipliers in magnitute.

If chosen less of equal to zero, no upper limit is imposed. Otherwise, the entries exceeding the given limit are set to the value closest to the limit.

Definition at line 77 of file IpWarmStartIterateInitializer.hpp.

6.191.4.7 `Number Ipopt::WarmStartIterateInitializer::warm_start_target_mu_ [private]`

Target values for the barrier parameter in warm start option.

Definition at line 80 of file IpWarmStartIterateInitializer.hpp.

6.191.4.8 `bool Ipopt::WarmStartIterateInitializer::warm_start_entire_iterate_ [private]`

Indicator for which method in the [NLP](#) should be used to get the warm start.

Definition at line 83 of file IpWarmStartIterateInitializer.hpp.

The documentation for this class was generated from the following file:

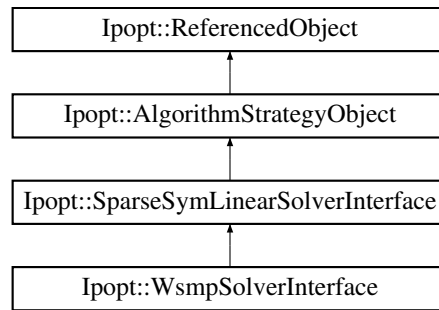
- [Algorithm/IpWarmStartIterateInitializer.hpp](#)

6.192 Ipopt::WsmvSolverInterface Class Reference

Interface to the linear solver Wsmv, derived from [SparseSymLinearSolverInterface](#).

```
#include <IpWsmvSolverInterface.hpp>
```

Inheritance diagram for Ipopt::WsmvSolverInterface:



Public Member Functions

- `bool InitializeImpl (const OptionsList &options, const std::string &prefix)`
overloaded from [AlgorithmStrategyObject](#)
- `virtual bool ProvidesDegeneracyDetection () const`
Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.
- `virtual ESymSolverStatus DetermineDependentRows (const Index *ia, const Index *ja, std::list< Index > &c_deps)`
This method determines the list of row indices of the linearly dependent rows.

Constructor/Destructor

- `WsmvSolverInterface ()`
Constructor.
- `virtual ~WsmvSolverInterface ()`
Destructor.

Methods for requesting solution of the linear system.

- `virtual ESymSolverStatus InitializeStructure (Index dim, Index nonzeros, const Index *ia, const Index *ja)`
Method for initializing internal structures.
- `virtual double * GetValuesArrayPtr ()`
Method returning an internal array into which the nonzero elements are to be stored.
- `virtual ESymSolverStatus MultiSolve (bool new_matrix, const Index *ia, const Index *ja, Index nrhs, double *rhs_vals, bool check_NegEVals, Index numberOfNegEVals)`
Solve operation for multiple right hand sides.
- `virtual Index NumberOfNegEVals () const`
Number of negative eigenvalues detected during last factorization.
- `virtual bool IncreaseQuality ()`
Request to increase quality of solution for next solve.

- virtual bool [ProvidesInertia](#) () const
Query whether inertia is computed by linear solver.
- [EMatrixFormat](#) [MatrixFormat](#) () const
Query of requested matrix type that the linear solver understands.

Static Public Member Functions

- static void [RegisterOptions](#) ([SmartPtr](#)< [RegisteredOptions](#) > roptions)
Methods for IpoptType.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [WsmpSolverInterface](#) (const [WsmpSolverInterface](#) &)
Copy Constructor.
- void [operator=](#) (const [WsmpSolverInterface](#) &)
Overloaded Equals Operator.

Internal functions

- [ESymSolverStatus](#) [SymbolicFactorization](#) (const [Index](#) *ia, const [Index](#) *ja)
Call Wsmp to do the analysis phase.
- [ESymSolverStatus](#) [InternalSymFact](#) (const [Index](#) *ia, const [Index](#) *ja, [Index](#) numberOfNegEVals)
Call Wsmp to really do the analysis phase.
- [ESymSolverStatus](#) [Factorization](#) (const [Index](#) *ia, const [Index](#) *ja, bool check_NegEVals, [Index](#) numberOfNegEVals)
Call Wsmp to factorize the [Matrix](#).
- [ESymSolverStatus](#) [Solve](#) (const [Index](#) *ia, const [Index](#) *ja, [Index](#) nrhs, double *rhs_vals)
Call Wsmpx to do the Solve.

Private Attributes

- [Index](#) [matrix_file_number_](#)
Counter for matrix file numbers.

Information about the matrix

- [Index](#) [dim_](#)
Number of rows and columns of the matrix.
- [Index](#) [nonzeros_](#)
Number of nonzeros of the matrix in triplet representation.
- double * [a_](#)
Array for storing the values of the matrix.

Solver specific options

- [Index](#) [wsmp_num_threads_](#)

- *Option that controls the matching strategy.*
- [Number wsm_solver_pivotol_](#)
Pivot tolerance.
- [Number wsm_solver_pivotolmax_](#)
Maximal pivot tolerance.
- [Index wsm_solver_scaling_](#)
Indicating which of WSMP's scaling methods should be used.
- [Number wsm_solver_singularity_threshold_](#)
WSMP's singularity threshold.
- [Index wsm_solver_write_matrix_iteration_](#)
iteration number in which matrices are to be written out
- [bool skip_inertia_check_](#)
Flag indicating if the inertia is always assumed to be correct.
- [bool wsm_solver_no_pivoting_](#)
Flag indicating whether the positive definite version of WSMP should be used.

Information about most recent factorization/solve

- [Index negevals_](#)
Number of negative eigenvalues.

Initialization flags

- [bool initialized_](#)
Flag indicating if internal data is initialized.
- [bool printed_num_threads_](#)
Flag indicating if we already printed how many threads are used by WSMP.
- [bool pivotol_changed_](#)
Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed, or the computation of the ordering has been triggered with DPARNM[14].
- [bool have_symbolic_factorization_](#)
Flag indicating whether symbolic factorization and order has already been performed.
- [Index factorizations_since_recomputed_ordering_](#)
Counter indicating how many factorizations have been done since the last recomputation of the ordering.

Solver specific information

- [ipfint * IPARM_](#)
Integer parameter array for WSSMP.
- [double * DPARM_](#)
Double precision parameter array for WSSMP.
- [ipfint * PERM_](#)
WSSMP's permutation vector.
- [ipfint * INV_P_](#)
WSSMP's inverse permutation vector.
- [ipfint * MRP_](#)
WSSMP's internal MRP array.

Additional Inherited Members

6.192.1 Detailed Description

Interface to the linear solver Wsm, derived from [SparseSymLinearSolverInterface](#).

For details, see description of [SparseSymLinearSolverInterface](#) base class.

Definition at line 24 of file IpWsmSolverInterface.hpp.

6.192.2 Constructor & Destructor Documentation

6.192.2.1 `Ipopt::WsmSolverInterface::WsmSolverInterface ()`

Constructor.

6.192.2.2 `virtual Ipopt::WsmSolverInterface::~~WsmSolverInterface () [virtual]`

Destructor.

6.192.2.3 `Ipopt::WsmSolverInterface::WsmSolverInterface (const WsmSolverInterface &) [private]`

Copy Constructor.

6.192.3 Member Function Documentation

6.192.3.1 `bool Ipopt::WsmSolverInterface::InitializeImpl (const OptionsList & options, const std::string & prefix) [virtual]`

overloaded from [AlgorithmStrategyObject](#)

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.192.3.2 `virtual ESymSolverStatus Ipopt::WsmSolverInterface::InitializeStructure (Index dim, Index nonzeros, const Index * ia, const Index * ja) [virtual]`

Method for initializing internal structures.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.192.3.3 `virtual double* Ipopt::WsmSolverInterface::GetValuesArrayPtr () [virtual]`

Method returning an internal array into which the nonzero elements are to be stored.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.192.3.4 `virtual ESymSolverStatus Ipopt::WsmSolverInterface::MultiSolve (bool new_matrix, const Index * ia, const Index * ja, Index nrhs, double * rhs_vals, bool check_NegEVals, Index numberOfNegEVals) [virtual]`

Solve operation for multiple right hand sides.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.192.3.5 `virtual Index Ipopt::WsmSolverInterface::NumberOfNegEVals () const [virtual]`

Number of negative eigenvalues detected during last factorization.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.192.3.6 `virtual bool Ipopt::WsmSolverInterface::IncreaseQuality () [virtual]`

Request to increase quality of solution for next solve.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

6.192.3.7 `virtual bool Ipopt::WsmSolverInterface::ProvidesInertia () const [inline], [virtual]`

Query whether inertia is computed by linear solver.

Returns true, if linear solver provides inertia.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 76 of file IpWsmSolverInterface.hpp.

6.192.3.8 **EMatrixFormat** Ipopt::WsmSolverInterface::MatrixFormat () const [inline], [virtual]

Query of requested matrix type that the linear solver understands.

Implements [Ipopt::SparseSymLinearSolverInterface](#).

Definition at line 83 of file IpWsmSolverInterface.hpp.

6.192.3.9 **static void** Ipopt::WsmSolverInterface::RegisterOptions (SmartPtr< RegisteredOptions > *roptions*) [static]

Methods for IpoptType.

6.192.3.10 **virtual bool** Ipopt::WsmSolverInterface::ProvidesDegeneracyDetection () const [virtual]

Query whether the indices of linearly dependent rows/columns can be determined by this linear solver.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

6.192.3.11 **virtual ESymSolverStatus** Ipopt::WsmSolverInterface::DetermineDependentRows (const Index * *ia*, const Index * *ja*, std::list< Index > & *c_deps*) [virtual]

This method determines the list of row indices of the linearly dependent rows.

Reimplemented from [Ipopt::SparseSymLinearSolverInterface](#).

6.192.3.12 **void** Ipopt::WsmSolverInterface::operator= (const WsmSolverInterface &) [private]

Overloaded Equals Operator.

6.192.3.13 **ESymSolverStatus** Ipopt::WsmSolverInterface::SymbolicFactorization (const Index * *ia*, const Index * *ja*) [private]

Call Wsm to do the analysis phase.

6.192.3.14 **ESymSolverStatus** Ipopt::WsmSolverInterface::InternalSymFact (const Index * *ia*, const Index * *ja*, Index *numberOfNegEvals*) [private]

Call Wsm to really do the analysis phase.

6.192.3.15 **ESymSolverStatus** Ipopt::WsmSolverInterface::Factorization (const Index * *ia*, const Index * *ja*, bool *check_NegEvals*, Index *numberOfNegEvals*) [private]

Call Wsm to factorize the [Matrix](#).

6.192.3.16 **ESymSolverStatus** Ipopt::WsmSolverInterface::Solve (const Index * *ia*, const Index * *ja*, Index *nrhs*, double * *rhs_vals*) [private]

Call Wsm to do the Solve.

6.192.4 Member Data Documentation

6.192.4.1 Index Ipopt::WsmpSolverInterface::dim_ [private]

Number of rows and columns of the matrix.

Definition at line 123 of file IpWsmpSolverInterface.hpp.

6.192.4.2 Index Ipopt::WsmpSolverInterface::nonzeros_ [private]

Number of nonzeros of the matrix in triplet representation.

Definition at line 126 of file IpWsmpSolverInterface.hpp.

6.192.4.3 double* Ipopt::WsmpSolverInterface::a_ [private]

Array for storing the values of the matrix.

Definition at line 129 of file IpWsmpSolverInterface.hpp.

6.192.4.4 Index Ipopt::WsmpSolverInterface::wsmp_num_threads_ [private]

Option that controls the matching strategy.

Definition at line 149 of file IpWsmpSolverInterface.hpp.

6.192.4.5 Number Ipopt::WsmpSolverInterface::wsmp_pivotol_ [private]

Pivot tolerance.

Definition at line 151 of file IpWsmpSolverInterface.hpp.

6.192.4.6 Number Ipopt::WsmpSolverInterface::wsmp_pivotolmax_ [private]

Maximal pivot tolerance.

Definition at line 153 of file IpWsmpSolverInterface.hpp.

6.192.4.7 Index Ipopt::WsmpSolverInterface::wsmp_scaling_ [private]

Indicating which of WSMP's scaling methods should be used.

Definition at line 155 of file IpWsmpSolverInterface.hpp.

6.192.4.8 Number Ipopt::WsmpSolverInterface::wsmp_singularity_threshold_ [private]

WSMP's singularity threshold.

The smaller this value the less likely a matrix is declared singular.

Definition at line 158 of file IpWsmpSolverInterface.hpp.

6.192.4.9 Index Ipopt::WsmpSolverInterface::wsmp_write_matrix_iteration_ [private]

iteration number in which matrices are to be written out

Definition at line 160 of file IpWsmpSolverInterface.hpp.

6.192.4.10 bool Ipopt::WsmpSolverInterface::skip_inertia_check_ [private]

Flag indicating if the inertia is always assumed to be correct.

Definition at line 163 of file IpWsmpSolverInterface.hpp.

6.192.4.11 `bool Ipopt::WsmSolverInterface::wsmp_no_pivoting_ [private]`

Flag indicating whether the positive definite version of WSMP should be used.

Definition at line 166 of file IpWsmSolverInterface.hpp.

6.192.4.12 `Index Ipopt::WsmSolverInterface::matrix_file_number_ [private]`

Counter for matrix file numbers.

Definition at line 170 of file IpWsmSolverInterface.hpp.

6.192.4.13 `Index Ipopt::WsmSolverInterface::negevals_ [private]`

Number of negative eigenvalues.

Definition at line 175 of file IpWsmSolverInterface.hpp.

6.192.4.14 `bool Ipopt::WsmSolverInterface::initialized_ [private]`

Flag indicating if internal data is initialized.

For initialization, this object needs to have seen a matrix

Definition at line 182 of file IpWsmSolverInterface.hpp.

6.192.4.15 `bool Ipopt::WsmSolverInterface::printed_num_threads_ [private]`

Flag indicating if we already printed how many threads are used by WSMP.

Definition at line 185 of file IpWsmSolverInterface.hpp.

6.192.4.16 `bool Ipopt::WsmSolverInterface::pivotl_changed_ [private]`

Flag indicating if the matrix has to be refactorized because the pivot tolerance has been changed, or the computation of the ordering has been triggered with DPARNM[14].

Definition at line 189 of file IpWsmSolverInterface.hpp.

6.192.4.17 `bool Ipopt::WsmSolverInterface::have_symbolic_factorization_ [private]`

Flag indicating whether symbolic factorization and order has already been performed.

Definition at line 192 of file IpWsmSolverInterface.hpp.

6.192.4.18 `Index Ipopt::WsmSolverInterface::factorizations_since_recomputed_ordering_ [private]`

Counter indicating how many factorizations have been done sine the last recomputation of the ordering.

Definition at line 195 of file IpWsmSolverInterface.hpp.

6.192.4.19 `ipfint* Ipopt::WsmSolverInterface::IPARM_ [private]`

Integer parameter array for WSSMP.

Definition at line 201 of file IpWsmSolverInterface.hpp.

6.192.4.20 `double* Ipopt::WsmSolverInterface::DPARM_ [private]`

Double precision parameter array for WSSMP.

Definition at line 203 of file IpWsmSolverInterface.hpp.

6.192.4.21 `ipfint* Ipopt::WsmSolverInterface::PERM_` [private]

WSSMP's permutation vector.

Definition at line 205 of file `IpWsmSolverInterface.hpp`.

6.192.4.22 `ipfint* Ipopt::WsmSolverInterface::INVP_` [private]

WSSMP's inverse permutation vector.

Definition at line 207 of file `IpWsmSolverInterface.hpp`.

6.192.4.23 `ipfint* Ipopt::WsmSolverInterface::MRP_` [private]

WSSMP's internal MRP array.

Definition at line 209 of file `IpWsmSolverInterface.hpp`.

The documentation for this class was generated from the following file:

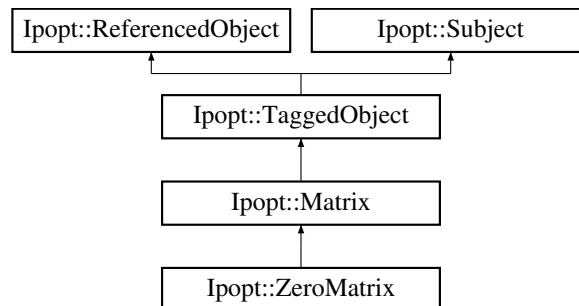
- Algorithm/LinearSolvers/[IpWsmSolverInterface.hpp](#)

6.193 `Ipopt::ZeroMatrix` Class Reference

Class for Matrices with only zero entries.

```
#include <IpZeroMatrix.hpp>
```

Inheritance diagram for `Ipopt::ZeroMatrix`:



Public Member Functions

Constructors / Destructors

- `ZeroMatrix` (const `MatrixSpace` *owner_space)
Constructor, taking the corresponding matrix space.
- `~ZeroMatrix` ()
Destructor.

Protected Member Functions

Methods overloaded from matrix

- virtual void `MultVectorImpl` (Number alpha, const `Vector` &x, Number beta, `Vector` &y) const

- Matrix-vector multiply.*
- virtual void [TransMultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix(transpose) vector multiply.
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &cols_norms, bool init) const
Compute the max-norm of the columns in the matrix.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ZeroMatrix](#) ()
Default Constructor.
- [ZeroMatrix](#) (const [ZeroMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [ZeroMatrix](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.193.1 Detailed Description

Class for Matrices with only zero entries.

Definition at line 20 of file IpZeroMatrix.hpp.

6.193.2 Constructor & Destructor Documentation

6.193.2.1 Ipopt::ZeroMatrix::ZeroMatrix (const [MatrixSpace](#) * owner_space)

Constructor, taking the corresponding matrix space.

6.193.2.2 Ipopt::ZeroMatrix::~~ZeroMatrix ()

Destructor.

6.193.2.3 Ipopt::ZeroMatrix::ZeroMatrix () [private]

Default Constructor.

6.193.2.4 Ipopt::ZeroMatrix::ZeroMatrix (const [ZeroMatrix](#) &) [private]

Copy Constructor.

6.193.3 Member Function Documentation

6.193.3.1 `virtual void Ipopt::ZeroMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const`
`[protected], [virtual]`

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.193.3.2 `virtual void Ipopt::ZeroMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y)`
`const [protected], [virtual]`

Matrix(transpose) vector multiply.

Computes $y = \alpha * \text{Matrix}^T * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.193.3.3 `virtual void Ipopt::ZeroMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const` `[inline],`
`[protected], [virtual]`

Compute the max-norm of the rows in the matrix.

The result is stored in *rows_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

Definition at line 44 of file `IpZeroMatrix.hpp`.

6.193.3.4 `virtual void Ipopt::ZeroMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const` `[inline],`
`[protected], [virtual]`

Compute the max-norm of the columns in the matrix.

The result is stored in *cols_norms*. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

Definition at line 47 of file `IpZeroMatrix.hpp`.

6.193.3.5 `virtual void Ipopt::ZeroMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory`
`category, const std::string & name, Index indent, const std::string & prefix) const` `[protected], [virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.193.3.6 `void Ipopt::ZeroMatrix::operator= (const ZeroMatrix &)` `[private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

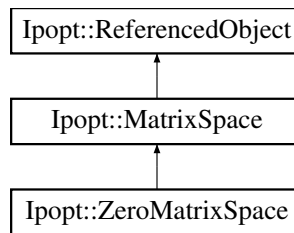
- [LinAlg/IpZeroMatrix.hpp](#)

6.194 Ipopt::ZeroMatrixSpace Class Reference

Class for matrix space for [ZeroMatrix](#).

```
#include <IpZeroMatrix.hpp>
```

Inheritance diagram for Ipopt::ZeroMatrixSpace:



Public Member Functions

- virtual [Matrix](#) * [MakeNew](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.
- [ZeroMatrix](#) * [MakeNewZeroMatrix](#) () const
Method for creating a new matrix of this specific type.

Constructors / Destructors

- [ZeroMatrixSpace](#) ([Index](#) nrows, [Index](#) ncols)
Constructor, given the number of row and columns.
- virtual [~ZeroMatrixSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ZeroMatrixSpace](#) ()
Default Constructor.
- [ZeroMatrixSpace](#) (const [ZeroMatrixSpace](#) &)
Copy Constructor.
- void [operator=](#) (const [ZeroMatrixSpace](#) &)
Overloaded Equals Operator.

6.194.1 Detailed Description

Class for matrix space for [ZeroMatrix](#).

Definition at line 79 of file IpZeroMatrix.hpp.

6.194.2 Constructor & Destructor Documentation

6.194.2.1 Ipopt::ZeroMatrixSpace::ZeroMatrixSpace ([Index](#) nrows, [Index](#) ncols) `[inline]`

Constructor, given the number of row and columns.

Definition at line 86 of file IpZeroMatrix.hpp.

6.194.2.2 `virtual Ipopt::ZeroMatrixSpace::~~ZeroMatrixSpace () [inline],[virtual]`

Destructor.

Definition at line 92 of file `IpZeroMatrix.hpp`.

6.194.2.3 `Ipopt::ZeroMatrixSpace::ZeroMatrixSpace () [private]`

Default Constructor.

6.194.2.4 `Ipopt::ZeroMatrixSpace::ZeroMatrixSpace (const ZeroMatrixSpace &) [private]`

Copy Constructor.

6.194.3 Member Function Documentation

6.194.3.1 `virtual Matrix* Ipopt::ZeroMatrixSpace::MakeNew () const [inline],[virtual]`

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Implements [Ipopt::MatrixSpace](#).

Definition at line 98 of file `IpZeroMatrix.hpp`.

6.194.3.2 `ZeroMatrix* Ipopt::ZeroMatrixSpace::MakeNewZeroMatrix () const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 104 of file `IpZeroMatrix.hpp`.

6.194.3.3 `void Ipopt::ZeroMatrixSpace::operator= (const ZeroMatrixSpace &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

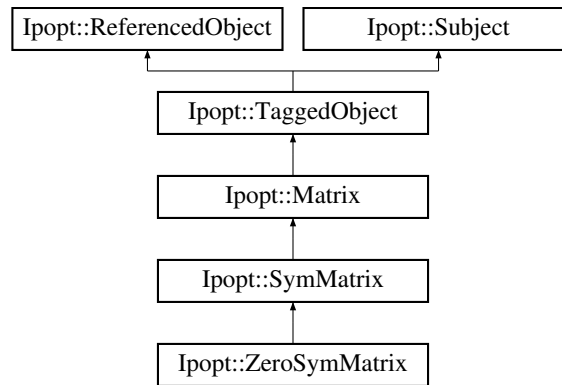
- [LinAlg/IpZeroMatrix.hpp](#)

6.195 Ipopt::ZeroSymMatrix Class Reference

Class for Symmetric Matrices with only zero entries.

```
#include <IpZeroSymMatrix.hpp>
```

Inheritance diagram for `Ipopt::ZeroSymMatrix`:



Public Member Functions

Constructors / Destructors

- [ZeroSymMatrix](#) (const [SymMatrixSpace](#) *owner_space)
Constructor, taking the corresponding matrix space.
- [~ZeroSymMatrix](#) ()
Destructor.

Protected Member Functions

Methods overloaded from matrix

- virtual void [MultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Matrix-vector multiply.
- virtual void [TransMultVectorImpl](#) ([Number](#) alpha, const [Vector](#) &x, [Number](#) beta, [Vector](#) &y) const
Since the matrix is symmetric, it is only necessary to implement the MultVectorImpl method in a class that inherits from this base class.
- virtual void [ComputeRowAMaxImpl](#) ([Vector](#) &rows_norms, bool init) const
Compute the max-norm of the rows in the matrix.
- virtual void [ComputeColAMaxImpl](#) ([Vector](#) &cols_norms, bool init) const
Since the matrix is symmetric, the row and column max norms are identical.
- virtual void [PrintImpl](#) (const [Journalist](#) &jnlst, [EJournalLevel](#) level, [EJournalCategory](#) category, const std::string &name, [Index](#) indent, const std::string &prefix) const
Print detailed information about the matrix.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ZeroSymMatrix](#) ()
Default Constructor.
- [ZeroSymMatrix](#) (const [ZeroSymMatrix](#) &)
Copy Constructor.
- void [operator=](#) (const [ZeroSymMatrix](#) &)
Overloaded Equals Operator.

Additional Inherited Members

6.195.1 Detailed Description

Class for Symmetric Matrices with only zero entries.

Definition at line 20 of file IpZeroSymMatrix.hpp.

6.195.2 Constructor & Destructor Documentation

6.195.2.1 Ipopt::ZeroSymMatrix::ZeroSymMatrix (const SymMatrixSpace * owner_space)

Constructor, taking the corresponding matrix space.

6.195.2.2 Ipopt::ZeroSymMatrix::~~ZeroSymMatrix ()

Destructor.

6.195.2.3 Ipopt::ZeroSymMatrix::ZeroSymMatrix () [private]

Default Constructor.

6.195.2.4 Ipopt::ZeroSymMatrix::ZeroSymMatrix (const ZeroSymMatrix &) [private]

Copy Constructor.

6.195.3 Member Function Documentation

6.195.3.1 virtual void Ipopt::ZeroSymMatrix::MultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]

Matrix-vector multiply.

Computes $y = \alpha * \text{Matrix} * x + \beta * y$

Implements [Ipopt::Matrix](#).

6.195.3.2 virtual void Ipopt::ZeroSymMatrix::TransMultVectorImpl (Number alpha, const Vector & x, Number beta, Vector & y) const [protected], [virtual]

Since the matrix is symmetric, it is only necessary to implement the MultVectorImpl method in a class that inherits from this base class.

If the TransMultVectorImpl is called, this base class automatically calls MultVectorImpl instead.

Reimplemented from [Ipopt::SymMatrix](#).

6.195.3.3 virtual void Ipopt::ZeroSymMatrix::ComputeRowAMaxImpl (Vector & rows_norms, bool init) const [inline], [protected], [virtual]

Compute the max-norm of the rows in the matrix.

The result is stored in rows_norms. The vector is assumed to be initialized.

Implements [Ipopt::Matrix](#).

Definition at line 44 of file IpZeroSymMatrix.hpp.

6.195.3.4 `virtual void Ipopt::ZeroSymMatrix::ComputeColAMaxImpl (Vector & cols_norms, bool init) const [inline], [protected], [virtual]`

Since the matrix is symmetric, the row and column max norms are identical.

Reimplemented from [Ipopt::SymMatrix](#).

Definition at line 47 of file `IpZeroSymMatrix.hpp`.

6.195.3.5 `virtual void Ipopt::ZeroSymMatrix::PrintImpl (const Journalist & jnlst, EJournalLevel level, EJournalCategory category, const std::string & name, Index indent, const std::string & prefix) const [protected], [virtual]`

Print detailed information about the matrix.

Implements [Ipopt::Matrix](#).

6.195.3.6 `void Ipopt::ZeroSymMatrix::operator= (const ZeroSymMatrix &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

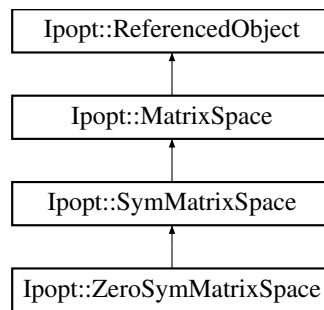
- [LinAlg/IpZeroSymMatrix.hpp](#)

6.196 Ipopt::ZeroSymMatrixSpace Class Reference

Class for matrix space for [ZeroSymMatrix](#).

```
#include <IpZeroSymMatrix.hpp>
```

Inheritance diagram for `Ipopt::ZeroSymMatrixSpace`:



Public Member Functions

- virtual [Matrix](#) * [MakeNew](#) () const
Overloaded MakeNew method for the [MatrixSpace](#) base class.
- virtual [SymMatrix](#) * [MakeNewSymMatrix](#) () const
Overloaded method from [SymMatrixSpace](#) base class.
- [ZeroSymMatrix](#) * [MakeNewZeroSymMatrix](#) () const
Method for creating a new matrix of this specific type.

Constructors / Destructors

- [ZeroSymMatrixSpace](#) ([Index](#) dim)

- *Constructor, given the number of row and columns.*
virtual [~ZeroSymMatrixSpace](#) ()
Destructor.

Private Member Functions

Default Compiler Generated Methods

(Hidden to avoid implicit creation/calling).

These methods are not implemented and we do not want the compiler to implement them for us, so we declare them private and do not define them. This ensures that they will not be implicitly created/called.

- [ZeroSymMatrixSpace](#) ()
Default Constructor.
- [ZeroSymMatrixSpace](#) (const [ZeroSymMatrixSpace](#) &)
Copy Constructor.
- void [operator=](#) (const [ZeroSymMatrixSpace](#) &)
Overloaded Equals Operator.

6.196.1 Detailed Description

Class for matrix space for [ZeroSymMatrix](#).

Definition at line 79 of file `IpZeroSymMatrix.hpp`.

6.196.2 Constructor & Destructor Documentation

6.196.2.1 `Ipopt::ZeroSymMatrixSpace::ZeroSymMatrixSpace (Index dim) [inline]`

Constructor, given the number of row and columns.

Definition at line 86 of file `IpZeroSymMatrix.hpp`.

6.196.2.2 `virtual Ipopt::ZeroSymMatrixSpace::~~ZeroSymMatrixSpace () [inline],[virtual]`

Destructor.

Definition at line 92 of file `IpZeroSymMatrix.hpp`.

6.196.2.3 `Ipopt::ZeroSymMatrixSpace::ZeroSymMatrixSpace () [private]`

Default Constructor.

6.196.2.4 `Ipopt::ZeroSymMatrixSpace::ZeroSymMatrixSpace (const ZeroSymMatrixSpace &) [private]`

Copy Constructor.

6.196.3 Member Function Documentation

6.196.3.1 `virtual Matrix* Ipopt::ZeroSymMatrixSpace::MakeNew () const [inline],[virtual]`

Overloaded MakeNew method for the [MatrixSpace](#) base class.

Reimplemented from [Ipopt::SymMatrixSpace](#).

Definition at line 98 of file `IpZeroSymMatrix.hpp`.

6.196.3.2 `virtual SymMatrix* Ipopt::ZeroSymMatrixSpace::MakeNewSymMatrix () const [inline],[virtual]`

Overloaded method from [SymMatrixSpace](#) base class.

Implements [Ipopt::SymMatrixSpace](#).

Definition at line 105 of file `IpZeroSymMatrix.hpp`.

6.196.3.3 `ZeroSymMatrix* Ipopt::ZeroSymMatrixSpace::MakeNewZeroSymMatrix () const [inline]`

Method for creating a new matrix of this specific type.

Definition at line 111 of file `IpZeroSymMatrix.hpp`.

6.196.3.4 `void Ipopt::ZeroSymMatrixSpace::operator= (const ZeroSymMatrixSpace &) [private]`

Overloaded Equals Operator.

The documentation for this class was generated from the following file:

- [LinAlg/IpZeroSymMatrix.hpp](#)

7 File Documentation

7.1 Algorithm/Inexact/IpInexactAlgBuilder.hpp File Reference

```
#include "IpAlgBuilder.hpp"
```

Classes

- class [Ipopt::InexactAlgorithmBuilder](#)
Builder to create a complete IpoptAlg object for the inexact step computation version.

Namespaces

- [Ipopt](#)

Functions

- void [Ipopt::AddInexactDefaultOptions](#) (OptionsList &options_list)
Function for setting options whos default is different for the inexact algorithm compared to the defaults for the regular Ipopt algorithm.

7.2 Algorithm/Inexact/IpInexactCq.hpp File Reference

```
#include "IpIpoptCalculatedQuantities.hpp"
#include "IpInexactData.hpp"
```

Classes

- class [Ipopt::InexactCq](#)
Class for all Chen-Goldfarb penalty method specific calculated quantities.

Namespaces

- [Ipopt](#)

7.3 Algorithm/Inexact/IpInexactData.hpp File Reference

```
#include "IpIpoptData.hpp"
```

Classes

- class [Ipopt::InexactData](#)
Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm.

Namespaces

- [Ipopt](#)

7.4 Algorithm/Inexact/IpInexactDoglegNormal.hpp File Reference

```
#include "IpInexactNormalStepCalc.hpp"  
#include "IpInexactNewtonNormal.hpp"  
#include "IpInexactNormalTerminationTester.hpp"
```

Classes

- class [Ipopt::InexactDoglegNormalStep](#)
Compute the normal step using a dogleg approach.

Namespaces

- [Ipopt](#)

7.5 Algorithm/Inexact/IpInexactLSAcceptor.hpp File Reference

```
#include "IpBacktrackingLSAcceptor.hpp"  
#include "IpInexactCq.hpp"
```

Classes

- class [Ipopt::InexactLSAcceptor](#)
Penalty function line search for the inexact step algorithm version.

Namespaces

- [Ipopt](#)

7.6 Algorithm/Inexact/IpInexactNewtonNormal.hpp File Reference

```
#include "IpAlgStrategy.hpp"
#include "IpAugSystemSolver.hpp"
#include "IpInexactCq.hpp"
```

Classes

- class [Ipopt::InexactNewtonNormalStep](#)
Compute the "Newton" normal step from the (slack-scaled) augmented system.

Namespaces

- [Ipopt](#)

7.7 Algorithm/Inexact/IpInexactNormalStepCalc.hpp File Reference

```
#include "IpAlgStrategy.hpp"
#include "IpInexactCq.hpp"
```

Classes

- class [Ipopt::InexactNormalStepCalculator](#)
Base class for computing the normal step for the inexact step calculation algorithm.

Namespaces

- [Ipopt](#)

7.8 Algorithm/Inexact/IpInexactNormalTerminationTester.hpp File Reference

```
#include "IpIterativeSolverTerminationTester.hpp"
```

Classes

- class [Ipopt::InexactNormalTerminationTester](#)

This class implements the termination tests for the primal-dual system.

Namespaces

- [Ipopt](#)

7.9 Algorithm/Inexact/IpInexactPDSolver.hpp File Reference

```
#include "IpAlgStrategy.hpp"
#include "IpAugSystemSolver.hpp"
#include "IpPDPerturbationHandler.hpp"
#include "IpInexactCq.hpp"
```

Classes

- class [Ipopt::InexactPDSolver](#)

This is the implementation of the Primal-Dual System, allowing the usage of an inexact linear solver.

Namespaces

- [Ipopt](#)

7.10 Algorithm/Inexact/IpInexactPDTerminationTester.hpp File Reference

```
#include "IpIterativeSolverTerminationTester.hpp"
```

Classes

- class [Ipopt::InexactPDTerminationTester](#)

This class implements the termination tests for the primal-dual system.

Namespaces

- [Ipopt](#)

7.11 Algorithm/Inexact/IpInexactRegOp.hpp File Reference

```
#include "IpSmartPtr.hpp"
```

Namespaces

- [Ipopt](#)

Functions

- void [Ipopt::RegisterOptions_Inexact](#) (const SmartPtr< RegisteredOptions > &options)

7.12 Algorithm/Inexact/IpInexactSearchDirCalc.hpp File Reference

```
#include "IpSearchDirCalculator.hpp"
#include "IpInexactCq.hpp"
#include "IpInexactNormalStepCalc.hpp"
#include "IpInexactPDSolver.hpp"
```

Classes

- class [Ipopt::InexactSearchDirCalculator](#)
Implementation of the search direction calculator that computes the search direction using iterative linear solvers.

Namespaces

- [Ipopt](#)

7.13 Algorithm/Inexact/IpInexactTSymScalingMethod.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpTSymScalingMethod.hpp"
#include "IpInexactCq.hpp"
```

Classes

- class [Ipopt::InexactTSymScalingMethod](#)
Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm.

Namespaces

- [Ipopt](#)

7.14 Algorithm/Inexact/IpIterativePardisoSolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
#include "IpInexactCq.hpp"
#include "IpIterativeSolverTerminationTester.hpp"
```

Classes

- class [Ipopt::IterativePardisoSolverInterface](#)
Interface to the linear solver Pardiso, derived from [SparseSymLinearSolverInterface](#).

Namespaces

- [Ipopt](#)

7.15 Algorithm/Inexact/IpIterativeSolverTerminationTester.hpp File Reference

```
#include "IpAlgStrategy.hpp"
#include "IpInexactCq.hpp"
```

Classes

- class [Ipopt::IterativeSolverTerminationTester](#)
This base class is for the termination tests for the iterative linear solver in the inexact version of [Ipopt](#).

Namespaces

- [Ipopt](#)

7.16 Algorithm/IpAdaptiveMuUpdate.hpp File Reference

```
#include "IpMuUpdate.hpp"
#include "IpLineSearch.hpp"
#include "IpMuOracle.hpp"
#include "IpFilter.hpp"
#include "IpQualityFunctionMuOracle.hpp"
```

Classes

- class [Ipopt::AdaptiveMuUpdate](#)
Non-monotone mu update.

Namespaces

- [Ipopt](#)

7.17 Algorithm/IpAlgBuilder.hpp File Reference

```
#include "IpIpoptAlg.hpp"
#include "IpReferenced.hpp"
#include "IpAugSystemSolver.hpp"
```

Classes

- class [Ipopt::AlgorithmBuilder](#)
Builder to create a complete IpoptAlg object.

Namespaces

- [Ipopt](#)

7.18 Algorithm/IpAlgorithmRegOp.hpp File Reference

```
#include "IpSmartPtr.hpp"
```

Namespaces

- [Ipopt](#)

Functions

- void [Ipopt::RegisterOptions_Algorithm](#) (const SmartPtr< RegisteredOptions > &options)

7.19 Algorithm/IpAlgStrategy.hpp File Reference

```
#include "IpOptionsList.hpp"  
#include "IpJournalist.hpp"  
#include "IpIpoptCalculatedQuantities.hpp"  
#include "IpIpoptNLP.hpp"  
#include "IpIpoptData.hpp"
```

Classes

- class [Ipopt::AlgorithmStrategyObject](#)
This is the base class for all algorithm strategy objects.

Namespaces

- [Ipopt](#)

7.20 Algorithm/IpAugRestoSystemSolver.hpp File Reference

```
#include "IpAugSystemSolver.hpp"
```

Classes

- class [Ipopt::AugRestoSystemSolver](#)

Class that converts the an augmented system with compound restoration pieces into a smaller "pivoted" system to be solved with an existing [AugSystemSolver](#).

Namespaces

- [Ipopt](#)

7.21 Algorithm/IpAugSystemSolver.hpp File Reference

```
#include "IpSymMatrix.hpp"
#include "IpSymLinearSolver.hpp"
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::AugSystemSolver](#)

Base class for Solver for the augmented system.

Namespaces

- [Ipopt](#)

Functions

- [Ipopt::DECLARE_STD_EXCEPTION](#) (FATAL_ERROR_IN_LINEAR_SOLVER)

7.22 Algorithm/IpBacktrackingLineSearch.hpp File Reference

```
#include "IpLineSearch.hpp"
#include "IpBacktrackingLSAcceptor.hpp"
#include "IpRestoPhase.hpp"
#include "IpConvCheck.hpp"
```

Classes

- class [Ipopt::BacktrackingLineSearch](#)

General implementation of a backtracking line search.

Namespaces

- [Ipopt](#)

7.23 Algorithm/lpBacktrackingLSAcceptor.hpp File Reference

```
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::BacktrackingLSAcceptor](#)
Base class for backtracking line search acceptors.

Namespaces

- [Ipopt](#)

7.24 Algorithm/lpConvCheck.hpp File Reference

```
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::ConvergenceCheck](#)
Base class for checking the algorithm termination criteria.

Namespaces

- [Ipopt](#)

7.25 Algorithm/lpDefaultIterateInitializer.hpp File Reference

```
#include "IpIterateInitializer.hpp"  
#include "IpEqMultCalculator.hpp"  
#include "IpAugSystemSolver.hpp"
```

Classes

- class [Ipopt::DefaultIterateInitializer](#)
Class implementing the default initialization procedure (based on user options) for the iterates.

Namespaces

- [Ipopt](#)

7.26 Algorithm/lpEqMultCalculator.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::EqMultiplierCalculator](#)
Base Class for objects that compute estimates for the equality constraint multipliers y_c and y_d .

Namespaces

- [Ipopt](#)

7.27 Algorithm/lpEquilibrationScaling.hpp File Reference

```
#include "IpNLPScaling.hpp"
#include "IpNLP.hpp"
```

Classes

- class [Ipopt::EquilibrationScaling](#)
This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point.
- class [Ipopt::PointPerturber](#)
This class is a simple object for generating randomly perturbed points that are within the [NLP](#) bounds.

Namespaces

- [Ipopt](#)

7.28 Algorithm/lpExactHessianUpdater.hpp File Reference

```
#include "IpHessianUpdater.hpp"
```

Classes

- class [Ipopt::ExactHessianUpdater](#)
Implementation of the [HessianUpdater](#) for the use of exact second derivatives.

Namespaces

- [Ipopt](#)

7.29 Algorithm/lpFilter.hpp File Reference

```
#include "IpJournalist.hpp"
#include "IpDebug.hpp"
#include <list>
#include <vector>
```

Classes

- class [Ipopt::FilterEntry](#)
Class for one filter entry.
- class [Ipopt::Filter](#)
Class for the filter.

Namespaces

- [Ipopt](#)

7.30 Algorithm/lpFilterLSAccepter.hpp File Reference

```
#include "IpFilter.hpp"
#include "IpBacktrackingLSAccepter.hpp"
#include "IpPDSystemSolver.hpp"
```

Classes

- class [Ipopt::FilterLSAccepter](#)
Filter line search.

Namespaces

- [Ipopt](#)

7.31 Algorithm/lpGenAugSystemSolver.hpp File Reference

```
#include "IpAugSystemSolver.hpp"
#include "IpGenKKTSolverInterface.hpp"
```

Classes

- class [Ipopt::GenAugSystemSolver](#)
Solver for the augmented system using GenKKTSolverInterfaces.

Namespaces

- [Ipopt](#)

7.32 Algorithm/IpGradientScaling.hpp File Reference

```
#include "IpNLPScaling.hpp"  
#include "IpNLP.hpp"
```

Classes

- class [Ipopt::GradientScaling](#)
This class does problem scaling by setting the scaling parameters based on the maximum of the gradient at the user provided initial point.

Namespaces

- [Ipopt](#)

7.33 Algorithm/IpHessianUpdater.hpp File Reference

```
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::HessianUpdater](#)
Abstract base class for objects responsible for updating the Hessian information.

Namespaces

- [Ipopt](#)

7.34 Algorithm/IpIpoptAlg.hpp File Reference

```
#include "IpIpoptNLP.hpp"  
#include "IpAlgStrategy.hpp"  
#include "IpSearchDirCalculator.hpp"  
#include "IpLineSearch.hpp"  
#include "IpMuUpdate.hpp"  
#include "IpConvCheck.hpp"  
#include "IpOptionsList.hpp"  
#include "IpIterateInitializer.hpp"  
#include "IpIterationOutput.hpp"  
#include "IpAlgTypes.hpp"  
#include "IpHessianUpdater.hpp"  
#include "IpEqMultCalculator.hpp"
```

Classes

- class [Ipopt::IpoptAlgorithm](#)
The main ipopt algorithm class.

Namespaces

- [Ipopt](#)

Functions

Exceptions

- [Ipopt::DECLARE_STD_EXCEPTION](#) (STEP_COMPUTATION_FAILED)

7.35 Algorithm/IpIpoptCalculatedQuantities.hpp File Reference

```
#include "IpSmartPtr.hpp"
#include "IpCachedResults.hpp"
#include <string>
```

Classes

- class [Ipopt::IpoptAdditionalCq](#)
Base class for additional calculated quantities that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers.
- class [Ipopt::IpoptCalculatedQuantities](#)
Class for all IPOPT specific calculated quantities.

Namespaces

- [Ipopt](#)

Enumerations

- enum [Ipopt::ENormType](#) { [Ipopt::NORM_1](#) =0, [Ipopt::NORM_2](#), [Ipopt::NORM_MAX](#) }
Norm types.

7.36 Algorithm/IpIpoptData.hpp File Reference

```
#include "IpSymMatrix.hpp"
#include "IpOptionsList.hpp"
#include "IpIteratesVector.hpp"
#include "IpRegOptions.hpp"
#include "IpTimingStatistics.hpp"
```

Classes

- class [Ipopt::IpoptAdditionalData](#)

Base class for additional data that is special to a particular type of algorithm, such as the CG penalty function, or using iterative linear solvers.

- class [Ipopt::IpoptData](#)

Class to organize all the data required by the algorithm.

Namespaces

- [Ipopt](#)

7.37 Algorithm/IpIpoptNLP.hpp File Reference

```
#include "IpNLP.hpp"
#include "IpJournalist.hpp"
#include "IpNLPScaling.hpp"
```

Classes

- class [Ipopt::IpoptNLP](#)

This is the abstract base class for classes that map the traditional [NLP](#) into something that is more useful by [Ipopt](#).

Namespaces

- [Ipopt](#)

7.38 Algorithm/IpIterateInitializer.hpp File Reference

```
#include "IpAlgStrategy.hpp"
#include "IpIpoptNLP.hpp"
#include "IpIpoptData.hpp"
#include "IpIpoptCalculatedQuantities.hpp"
```

Classes

- class [Ipopt::IterateInitializer](#)

Base class for all methods for initializing the iterates.

Namespaces

- [Ipopt](#)

7.39 Algorithm/IpIteratesVector.hpp File Reference

```
#include "IpCompoundVector.hpp"
```

Classes

- class [Ipopt::IteratesVector](#)
Specialized [CompoundVector](#) class specifically for the algorithm iterates.
- class [Ipopt::IteratesVectorSpace](#)
[Vector](#) Space for the [IteratesVector](#) class.

Namespaces

- [Ipopt](#)

7.40 Algorithm/IpIterationOutput.hpp File Reference

```
#include "IpAlgStrategy.hpp"  
#include "IpIpoptNLP.hpp"  
#include "IpIpoptData.hpp"  
#include "IpIpoptCalculatedQuantities.hpp"
```

Classes

- class [Ipopt::IterationOutput](#)
Base class for objects that do the output summary per iteration.

Namespaces

- [Ipopt](#)

7.41 Algorithm/IpLeastSquareMults.hpp File Reference

```
#include "IpAugSystemSolver.hpp"  
#include "IpEqMultCalculator.hpp"
```

Classes

- class [Ipopt::LeastSquareMultipliers](#)
Class for calculator for the least-square equality constraint multipliers.

Namespaces

- [Ipopt](#)

7.42 Algorithm/IpLimMemQuasiNewtonUpdater.hpp File Reference

```
#include "IpHessianUpdater.hpp"
#include "IpLowRankUpdateSymMatrix.hpp"
#include "IpMultiVectorMatrix.hpp"
#include "IpDenseVector.hpp"
#include "IpDenseGenMatrix.hpp"
#include "IpDenseSymMatrix.hpp"
```

Classes

- class [Ipopt::LimMemQuasiNewtonUpdater](#)
Implementation of the [HessianUpdater](#) for limit-memory quasi-Newton approximation of the Lagrangian Hessian.

Namespaces

- [Ipopt](#)

7.43 Algorithm/IpLineSearch.hpp File Reference

```
#include "IpAlgStrategy.hpp"
#include "IpIpoptCalculatedQuantities.hpp"
```

Classes

- class [Ipopt::LineSearch](#)
Base class for line search objects.

Namespaces

- [Ipopt](#)

7.44 Algorithm/IpLoqoMuOracle.hpp File Reference

```
#include "IpMuOracle.hpp"
```

Classes

- class [Ipopt::LoqoMuOracle](#)
Implementation of the LOQO formula for computing the barrier parameter.

Namespaces

- [Ipopt](#)

7.45 Algorithm/IpLowRankAugSystemSolver.hpp File Reference

```
#include "IpAugSystemSolver.hpp"
#include "IpDenseGenMatrix.hpp"
#include "IpMultiVectorMatrix.hpp"
#include "IpDiagMatrix.hpp"
```

Classes

- class [Ipopt::LowRankAugSystemSolver](#)
Solver for the augmented system with [LowRankUpdateSymMatrix](#) Hessian matrices.

Namespaces

- [Ipopt](#)

7.46 Algorithm/IpLowRankSSAugSystemSolver.hpp File Reference

```
#include "IpAugSystemSolver.hpp"
#include "IpDiagMatrix.hpp"
#include "IpCompoundMatrix.hpp"
#include "IpCompoundVector.hpp"
#include "IpExpandedMultiVectorMatrix.hpp"
```

Classes

- class [Ipopt::LowRankSSAugSystemSolver](#)
Solver for the augmented system with [LowRankUpdateSymMatrix](#) Hessian matrices.

Namespaces

- [Ipopt](#)

7.47 Algorithm/IpMonotoneMuUpdate.hpp File Reference

```
#include "IpMuUpdate.hpp"
#include "IpLineSearch.hpp"
#include "IpRegOptions.hpp"
```

Classes

- class [Ipopt::MonotoneMuUpdate](#)
Monotone Mu Update.

Namespaces

- [Ipopt](#)

7.48 Algorithm/IpMuOracle.hpp File Reference

```
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::MuOracle](#)
Abstract Base Class for classes that are able to compute a suggested value of the barrier parameter that can be used as an oracle in the NonmontoneMuUpdate class.

Namespaces

- [Ipopt](#)

7.49 Algorithm/IpMuUpdate.hpp File Reference

```
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::MuUpdate](#)
Abstract Base Class for classes that implement methods for computing the barrier and fraction-to-the-boundary rule parameter for the current iteration.

Namespaces

- [Ipopt](#)

7.50 Algorithm/IpNLPBoundsRemover.hpp File Reference

```
#include "IpNLP.hpp"
```

Classes

- class [Ipopt::NLPBoundsRemover](#)
This is an adapter for an [NLP](#) that converts variable bound constraints to inequality constraints.

Namespaces

- [Ipopt](#)

7.51 Algorithm/lpNLPScaling.hpp File Reference

```
#include "IpOptionsList.hpp"
#include "IpRegOptions.hpp"
```

Classes

- class [lpopt::NLPScalingObject](#)
This is the abstract base class for problem scaling.
- class [lpopt::StandardScalingBase](#)
This is a base class for many standard scaling techniques.
- class [lpopt::NoNLPScalingObject](#)
Class implementing the scaling object that doesn't to any scaling.

Namespaces

- [lpopt](#)

7.52 Algorithm/lpOptErrorConvCheck.hpp File Reference

```
#include "IpConvCheck.hpp"
```

Classes

- class [lpopt::OptimalityErrorConvergenceCheck](#)
Brief Class Description.

Namespaces

- [lpopt](#)

7.53 Algorithm/lpOriglpoptNLP.hpp File Reference

```
#include "IpIpoptNLP.hpp"
#include "IpException.hpp"
#include "IpTimingStatistics.hpp"
```

Classes

- class [lpopt::OriglpoptNLP](#)
This class maps the traditional [NLP](#) into something that is more useful by [lpopt](#).

Namespaces

- [lpopt](#)

Enumerations

- enum [Ipopt::HessianApproximationType](#) { [Ipopt::EXACT](#) =0, [Ipopt::LIMITED_MEMORY](#) }
enumeration for the Hessian information type.
- enum [Ipopt::HessianApproximationSpace](#) { [Ipopt::NONLINEAR_VARS](#) =0, [Ipopt::ALL_VARS](#) }
enumeration for the Hessian approximation space.

7.54 Algorithm/IpOrigIterationOutput.hpp File Reference

```
#include "IpIterationOutput.hpp"
```

Classes

- class [Ipopt::OrigIterationOutput](#)
Class for the iteration summary output for the original [NLP](#).

Namespaces

- [Ipopt](#)

7.55 Algorithm/IpPDFullSpaceSolver.hpp File Reference

```
#include "IpPDSystemSolver.hpp"  
#include "IpAugSystemSolver.hpp"  
#include "IpPDPerturbationHandler.hpp"
```

Classes

- class [Ipopt::PDFullSpaceSolver](#)
This is the implemetation of the Primal-Dual System, using the full space approach with a direct linear solver.

Namespaces

- [Ipopt](#)

7.56 Algorithm/IpPDPerturbationHandler.hpp File Reference

```
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::PDPerturbationHandler](#)
Class for handling the perturbation factors δ_x , δ_s , δ_c , and δ_d in the primal dual system.

Namespaces

- [lpopt](#)

7.57 Algorithm/lpPDSearchDirCalc.hpp File Reference

```
#include "IpSearchDirCalculator.hpp"
#include "IpPDSystemSolver.hpp"
```

Classes

- class [lpopt::PDSearchDirCalculator](#)

Implementation of the search direction calculator that computes the pure primal dual step for the current barrier parameter.

Namespaces

- [lpopt](#)

7.58 Algorithm/lpPDSystemSolver.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
#include "IpAlgStrategy.hpp"
#include "IpIteratesVector.hpp"
```

Classes

- class [lpopt::PDSystemSolver](#)

Pure Primal Dual System Solver Base Class.

Namespaces

- [lpopt](#)

7.59 Algorithm/lpPenaltyLSAcceptor.hpp File Reference

```
#include "IpBacktrackingLSAcceptor.hpp"
#include "IpPDSystemSolver.hpp"
```

Classes

- class [lpopt::PenaltyLSAcceptor](#)

Penalty function line search.

Namespaces

- [Ipopt](#)

7.60 Algorithm/IpProbingMuOracle.hpp File Reference

```
#include "IpMuOracle.hpp"
#include "IpPDSystemSolver.hpp"
```

Classes

- class [Ipopt::ProbingMuOracle](#)
Implementation of the probing strategy for computing the barrier parameter.

Namespaces

- [Ipopt](#)

7.61 Algorithm/IpQualityFunctionMuOracle.hpp File Reference

```
#include "IpMuOracle.hpp"
#include "IpPDSystemSolver.hpp"
#include "IpIpoptCalculatedQuantities.hpp"
```

Classes

- class [Ipopt::QualityFunctionMuOracle](#)
Implementation of the probing strategy for computing the barrier parameter.

Namespaces

- [Ipopt](#)

7.62 Algorithm/IpRestoConvCheck.hpp File Reference

```
#include "IpOptErrorConvCheck.hpp"
#include "IpBacktrackingLSAcceptor.hpp"
```

Classes

- class [Ipopt::RestoConvergenceCheck](#)
Convergence check for the restoration phase.

Namespaces

- [Ipopt](#)

7.63 Algorithm/IpRestoFilterConvCheck.hpp File Reference

```
#include "IpRestoConvCheck.hpp"
#include "IpFilterLSAcceptor.hpp"
```

Classes

- class [Ipopt::RestoFilterConvergenceCheck](#)

This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism.

Namespaces

- [Ipopt](#)

7.64 Algorithm/IpRestIpoptNLP.hpp File Reference

```
#include "IpIpoptNLP.hpp"
#include "IpIpoptData.hpp"
#include "IpIpoptCalculatedQuantities.hpp"
#include "IpCompoundMatrix.hpp"
#include "IpCompoundSymMatrix.hpp"
#include "IpCompoundVector.hpp"
#include "IpIdentityMatrix.hpp"
#include "IpDiagMatrix.hpp"
#include "IpZeroMatrix.hpp"
#include "IpOrigIpoptNLP.hpp"
```

Classes

- class [Ipopt::RestIpoptNLP](#)

This class maps the traditional [NLP](#) into something that is more useful by [Ipopt](#).

Namespaces

- [Ipopt](#)

7.65 Algorithm/IpRestIterateInitializer.hpp File Reference

```
#include "IpIterateInitializer.hpp"
#include "IpEqMultCalculator.hpp"
```

Classes

- class [Ipopt::RestoIterateInitializer](#)

Class implementing the default initialization procedure (based on user options) for the iterates.

Namespaces

- [Ipopt](#)

7.66 Algorithm/IpRestoIterationOutput.hpp File Reference

```
#include "IpIterationOutput.hpp"
#include "IpOrigIterationOutput.hpp"
```

Classes

- class [Ipopt::RestoIterationOutput](#)

Class for the iteration summary output for the restoration phase.

Namespaces

- [Ipopt](#)

7.67 Algorithm/IpRestoMinC_1Nrm.hpp File Reference

```
#include "IpRestoPhase.hpp"
#include "IpIpoptAlg.hpp"
#include "IpEqMultCalculator.hpp"
```

Classes

- class [Ipopt::MinC_1NrmRestorationPhase](#)

Restoration Phase that minimizes the 1-norm of the constraint violation - using the interior point method ([Ipopt](#)).

Namespaces

- [Ipopt](#)

7.68 Algorithm/IpRestoPenaltyConvCheck.hpp File Reference

```
#include "IpRestoConvCheck.hpp"
#include "IpPenaltyLSAcceptor.hpp"
```


Classes

- class [Ipopt::RestoPenaltyConvergenceCheck](#)

This is the implementation of the restoration convergence check is the original algorithm used the filter globalization mechanism.

Namespaces

- [Ipopt](#)

7.69 Algorithm/IpRestoPhase.hpp File Reference

```
#include "IpAlgStrategy.hpp"
#include "IpIpoptNLP.hpp"
#include "IpIpoptData.hpp"
#include "IpIpoptCalculatedQuantities.hpp"
```

Classes

- class [Ipopt::RestorationPhase](#)

Base class for different restoration phases.

Namespaces

- [Ipopt](#)

Functions

Exceptions

- [Ipopt::DECLARE_STD_EXCEPTION](#) (RESTORATION_CONVERGED_TO_FEASIBLE_POINT)
Exception RESTORATION_FAILED for all trouble with the restoration phase.
- [Ipopt::DECLARE_STD_EXCEPTION](#) (RESTORATION_FAILED)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (RESTORATION_MAXITER_EXCEEDED)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (RESTORATION_CPUTIME_EXCEEDED)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (RESTORATION_USER_STOP)

7.70 Algorithm/IpRestoRestoPhase.hpp File Reference

```
#include "IpRestoPhase.hpp"
#include "IpIpoptAlg.hpp"
#include "IpEqMultCalculator.hpp"
```

Classes

- class [Ipopt::RestoRestorationPhase](#)

Recursive Restoration Phase for the MinC_1NrmRestorationPhase.

Namespaces

- [Ipopt](#)

7.71 Algorithm/IpSearchDirCalculator.hpp File Reference

```
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::SearchDirectionCalculator](#)
Base class for computing the search direction for the line search.

Namespaces

- [Ipopt](#)

7.72 Algorithm/IpStdAugSystemSolver.hpp File Reference

```
#include "IpAugSystemSolver.hpp"  
#include "IpCompoundMatrix.hpp"  
#include "IpCompoundSymMatrix.hpp"  
#include "IpCompoundVector.hpp"  
#include "IpSumSymMatrix.hpp"  
#include "IpDiagMatrix.hpp"  
#include "IpIdentityMatrix.hpp"
```

Classes

- class [Ipopt::StdAugSystemSolver](#)
Solver for the augmented system for triple type matrices.

Namespaces

- [Ipopt](#)

7.73 Algorithm/IpTimingStatistics.hpp File Reference

```
#include "IpReferenced.hpp"  
#include "IpJournalist.hpp"  
#include "IpTimedTask.hpp"
```

Classes

- class [Ipopt::TimingStatistics](#)

This class collects all timing statistics for [Ipopt](#).

Namespaces

- [Ipopt](#)

7.74 Algorithm/IpUserScaling.hpp File Reference

```
#include "IpNLPScaling.hpp"  
#include "IpNLP.hpp"
```

Classes

- class [Ipopt::UserScaling](#)

This class does problem scaling by getting scaling parameters from the user (through the [NLP](#) interface).

Namespaces

- [Ipopt](#)

7.75 Algorithm/IpWarmStartIterateInitializer.hpp File Reference

```
#include "IpIterateInitializer.hpp"  
#include "IpEqMultCalculator.hpp"
```

Classes

- class [Ipopt::WarmStartIterateInitializer](#)

Class implementing an initialization procedure for warm starts.

Namespaces

- [Ipopt](#)

7.76 Algorithm/LinearSolvers/hsl_ma77d.h File Reference

Classes

- struct [ma77_control_d](#)
- struct [ma77_info_d](#)

Macros

- `#define ma77_control ma77_control_d`
- `#define ma77_info ma77_info_d`
- `#define ma77_default_control ma77_default_control_d`
- `#define ma77_open_nelt ma77_open_nelt_d`
- `#define ma77_open ma77_open_d`
- `#define ma77_input_vars ma77_input_vars_d`
- `#define ma77_input_reals ma77_input_reals_d`
- `#define ma77_analyse ma77_analyse_d`
- `#define ma77_factor ma77_factor_d`
- `#define ma77_factor_solve ma77_factor_solve_d`
- `#define ma77_solve ma77_solve_d`
- `#define ma77_resid ma77_resid_d`
- `#define ma77_scale ma77_scale_d`
- `#define ma77_enquire_posdef ma77_enquire_posdef_d`
- `#define ma77_enquire_indef ma77_enquire_indef_d`
- `#define ma77_alter ma77_alter_d`
- `#define ma77_restart ma77_restart_d`
- `#define ma77_finalise ma77_finalise_d`
- `#define ma77_solve_fredholm ma77_solve_fredholm_d`
- `#define ma77_multiply ma77_multiply_d`

Typedefs

- `typedef double ma77pkgtype_d_`

Functions

- `void ma77_default_control_d (struct ma77_control_d *control)`
- `void ma77_open_nelt (const int n, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, const int nelt)`
- `void ma77_open_d (const int n, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`
- `void ma77_input_vars (const int idx, const int nvar, const int list[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`
- `void ma77_input_reals_d (const int idx, const int length, const ma77pkgtype_d_ reals[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`
- `void ma77_analyse (const int order[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`
- `void ma77_factor_d (const int posdef, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, const ma77pkgtype_d_ *scale)`
- `void ma77_factor_solve_d (const int posdef, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, const ma77pkgtype_d_ *scale, const int nrhs, const int lx, ma77pkgtype_d_ rhs[])`
- `void ma77_solve_d (const int job, const int nrhs, const int lx, ma77pkgtype_d_ x[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, const ma77pkgtype_d_ *scale)`
- `void ma77_resid_d (const int nrhs, const int lx, const ma77pkgtype_d_ x[], const int lresid, ma77pkgtype_d_ resid[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, ma77pkgtype_d_ *anorm_bnd)`
- `void ma77_scale_d (ma77pkgtype_d_ scale[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, ma77pkgtype_d_ *anorm)`

- void [ma77_enquire_posdef_d](#) (ma77pkgtype_d d[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- void [ma77_enquire_indef_d](#) (int piv_order[], [ma77pkgtype_d](#) d[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- void [ma77_alter_d](#) (const [ma77pkgtype_d](#) d[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- void [ma77_restart_d](#) (const char *restart_file, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- void [ma77_solve_fredholm_d](#) (int nrhs, int flag_out[], int lx, [ma77pkgtype_d](#) x[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info, const [ma77pkgtype_d](#) *scale)
- void [ma77_multiply_d](#) (int trans, int k, int lx, [ma77pkgtype_d](#) x[], int ly, [ma77pkgtype_d](#) y[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info, const [ma77pkgtype_d](#) *scale)
- void [ma77_finalise_d](#) (void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)

7.76.1 Macro Definition Documentation

7.76.1.1 #define ma77_control ma77_control_d

Definition at line 13 of file hsl_ma77d.h.

7.76.1.2 #define ma77_info ma77_info_d

Definition at line 14 of file hsl_ma77d.h.

7.76.1.3 #define ma77_default_control ma77_default_control_d

Definition at line 15 of file hsl_ma77d.h.

7.76.1.4 #define ma77_open_nelt ma77_open_nelt_d

Definition at line 16 of file hsl_ma77d.h.

7.76.1.5 #define ma77_open ma77_open_d

Definition at line 17 of file hsl_ma77d.h.

7.76.1.6 #define ma77_input_vars ma77_input_vars_d

Definition at line 18 of file hsl_ma77d.h.

7.76.1.7 #define ma77_input_reals ma77_input_reals_d

Definition at line 19 of file hsl_ma77d.h.

7.76.1.8 #define ma77_analyse ma77_analyse_d

Definition at line 20 of file hsl_ma77d.h.

7.76.1.9 #define ma77_factor ma77_factor_d

Definition at line 21 of file hsl_ma77d.h.

7.76.1.10 #define ma77_factor_solve ma77_factor_solve_d

Definition at line 22 of file hsl_ma77d.h.

7.76.1.11 #define ma77_solve ma77_solve_d

Definition at line 23 of file hsl_ma77d.h.

7.76.1.12 #define ma77_resid ma77_resid_d

Definition at line 24 of file hsl_ma77d.h.

7.76.1.13 #define ma77_scale ma77_scale_d

Definition at line 25 of file hsl_ma77d.h.

7.76.1.14 #define ma77_enquire_posdef ma77_enquire_posdef_d

Definition at line 26 of file hsl_ma77d.h.

7.76.1.15 #define ma77_enquire_indef ma77_enquire_indef_d

Definition at line 27 of file hsl_ma77d.h.

7.76.1.16 #define ma77_alter ma77_alter_d

Definition at line 28 of file hsl_ma77d.h.

7.76.1.17 #define ma77_restart ma77_restart_d

Definition at line 29 of file hsl_ma77d.h.

7.76.1.18 #define ma77_finalise ma77_finalise_d

Definition at line 30 of file hsl_ma77d.h.

7.76.1.19 #define ma77_solve_fredholm ma77_solve_fredholm_d

Definition at line 31 of file hsl_ma77d.h.

7.76.1.20 #define ma77_lmultiply ma77_lmultiply_d

Definition at line 32 of file hsl_ma77d.h.

7.76.2 Typedef Documentation**7.76.2.1 typedef double ma77pkgtype_d_**

Definition at line 35 of file hsl_ma77d.h.

7.76.3 Function Documentation**7.76.3.1 void ma77_default_control_d (struct ma77_control_d * control)**

7.76.3.2 void ma77_open_nelt (const int *n*, const char * *fname1*, const char * *fname2*, const char * *fname3*, const char * *fname4*, void ** *keep*, const struct ma77_control_d * *control*, struct ma77_info_d * *info*, const int *nelt*)

7.76.3.3 void ma77_open_d (const int *n*, const char * *fname1*, const char * *fname2*, const char * *fname3*, const char * *fname4*, void ** *keep*, const struct ma77_control_d * *control*, struct ma77_info_d * *info*)

- 7.76.3.4 void `ma77_input_vars` (const int *idx*, const int *nvar*, const int *list*[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*)
- 7.76.3.5 void `ma77_input_reals_d` (const int *idx*, const int *length*, const `ma77pkgtype_d_reals`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*)
- 7.76.3.6 void `ma77_analyse` (const int *order*[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*)
- 7.76.3.7 void `ma77_factor_d` (const int *posdef*, void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*, const `ma77pkgtype_d` * *scale*)
- 7.76.3.8 void `ma77_factor_solve_d` (const int *posdef*, void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*, const `ma77pkgtype_d` * *scale*, const int *nrhs*, const int *lx*, `ma77pkgtype_d_rhs`[])
- 7.76.3.9 void `ma77_solve_d` (const int *job*, const int *nrhs*, const int *lx*, `ma77pkgtype_d_x`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*, const `ma77pkgtype_d` * *scale*)
- 7.76.3.10 void `ma77_resid_d` (const int *nrhs*, const int *lx*, const `ma77pkgtype_d_x`[], const int *lresid*, `ma77pkgtype_d_resid`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*, `ma77pkgtype_d` * *anorm_bnd*)
- 7.76.3.11 void `ma77_scale_d` (`ma77pkgtype_d_scale`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*, `ma77pkgtype_d` * *anorm*)
- 7.76.3.12 void `ma77_enquire_posdef_d` (`ma77pkgtype_d_d`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*)
- 7.76.3.13 void `ma77_enquire_indef_d` (int *piv_order*[], `ma77pkgtype_d_d`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*)
- 7.76.3.14 void `ma77_alter_d` (const `ma77pkgtype_d_d`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*)
- 7.76.3.15 void `ma77_restart_d` (const char * *restart_file*, const char * *fname1*, const char * *fname2*, const char * *fname3*, const char * *fname4*, void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*)
- 7.76.3.16 void `ma77_solve_fredholm_d` (int *nrhs*, int *flag_out*[], int *lx*, `ma77pkgtype_d_x`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*, const `ma77pkgtype_d` * *scale*)
- 7.76.3.17 void `ma77_multiply_d` (int *trans*, int *k*, int *lx*, `ma77pkgtype_d_x`[], int *ly*, `ma77pkgtype_d_y`[], void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*, const `ma77pkgtype_d` * *scale*)
- 7.76.3.18 void `ma77_finalise_d` (void ** *keep*, const struct `ma77_control_d` * *control*, struct `ma77_info_d` * *info*)

7.77 Algorithm/LinearSolvers/hsl_ma86d.h File Reference

Classes

- struct [ma86_control_d](#)
- struct [ma86_info_d](#)

Macros

- #define [ma86_control](#) [ma86_control_d](#)

- `#define ma86_info ma86_info_d`
- `#define ma86_default_control ma86_default_control_d`
- `#define ma86_analyse ma86_analyse_d`
- `#define ma86_factor ma86_factor_d`
- `#define ma86_factor_solve ma86_factor_solve_d`
- `#define ma86_solve ma86_solve_d`
- `#define ma86_finalise ma86_finalise_d`

Typedefs

- `typedef double ma86pkgtype_d_`
- `typedef double ma86realtype_d_`

Functions

- `void ma86_default_control_d (struct ma86_control_d *control)`
- `void ma86_analyse_d (const int n, const int ptr[], const int row[], int order[], void **keep, const struct ma86_control_d *control, struct ma86_info_d *info)`
- `void ma86_factor_d (const int n, const int ptr[], const int row[], const ma86pkgtype_d_val[], const int order[], void **keep, const struct ma86_control_d *control, struct ma86_info_d *info, const ma86realtype_d_scale[])`
- `void ma86_factor_solve_d (const int n, const int ptr[], const int row[], const ma86pkgtype_d_val[], const int order[], void **keep, const struct ma86_control_d *control, struct ma86_info_d *info, const int nrhs, const int ldx, ma86pkgtype_d_x[], const ma86realtype_d_scale[])`
- `void ma86_solve_d (const int job, const int nrhs, const int ldx, ma86pkgtype_d_x, const int order[], void **keep, const struct ma86_control_d *control, struct ma86_info_d *info, const ma86realtype_d_scale[])`
- `void ma86_finalise_d (void **keep, const struct ma86_control_d *control)`

7.77.1 Macro Definition Documentation

7.77.1.1 `#define ma86_control ma86_control_d`

Definition at line 13 of file hsl_ma86d.h.

7.77.1.2 `#define ma86_info ma86_info_d`

Definition at line 14 of file hsl_ma86d.h.

7.77.1.3 `#define ma86_default_control ma86_default_control_d`

Definition at line 15 of file hsl_ma86d.h.

7.77.1.4 `#define ma86_analyse ma86_analyse_d`

Definition at line 16 of file hsl_ma86d.h.

7.77.1.5 `#define ma86_factor ma86_factor_d`

Definition at line 17 of file hsl_ma86d.h.

7.77.1.6 `#define ma86_factor_solve ma86_factor_solve_d`

Definition at line 18 of file hsl_ma86d.h.

7.77.1.7 `#define ma86_solve ma86_solve_d`

Definition at line 19 of file hsl_ma86d.h.

7.77.1.8 `#define ma86_finalise ma86_finalise_d`

Definition at line 20 of file hsl_ma86d.h.

7.77.2 Typedef Documentation

7.77.2.1 `typedef double ma86pkgtype_d_`

Definition at line 23 of file hsl_ma86d.h.

7.77.2.2 `typedef double ma86realtype_d_`

Definition at line 24 of file hsl_ma86d.h.

7.77.3 Function Documentation

7.77.3.1 `void ma86_default_control_d (struct ma86_control_d * control)`7.77.3.2 `void ma86_analyse_d (const int n, const int ptr[], const int row[], int order[], void ** keep, const struct ma86_control_d * control, struct ma86_info_d * info)`7.77.3.3 `void ma86_factor_d (const int n, const int ptr[], const int row[], const ma86pkgtype_d_val[], const int order[], void ** keep, const struct ma86_control_d * control, struct ma86_info_d * info, const ma86realtype_d_scale[])`7.77.3.4 `void ma86_factor_solve_d (const int n, const int ptr[], const int row[], const ma86pkgtype_d_val[], const int order[], void ** keep, const struct ma86_control_d * control, struct ma86_info_d * info, const int nrhs, const int idx, ma86pkgtype_d_x[], const ma86realtype_d_scale[])`7.77.3.5 `void ma86_solve_d (const int job, const int nrhs, const int idx, ma86pkgtype_d_x, const int order[], void ** keep, const struct ma86_control_d * control, struct ma86_info_d * info, const ma86realtype_d_scale[])`7.77.3.6 `void ma86_finalise_d (void ** keep, const struct ma86_control_d * control)`

7.78 Algorithm/LinearSolvers/hsl_ma97d.h File Reference

Classes

- struct [ma97_control_d](#)
- struct [ma97_info](#)

Macros

- `#define` [ma97_control](#) [ma97_control_d](#)
- `#define` [ma97_info](#) [ma97_info_d](#)
- `#define` [ma97_default_control](#) [ma97_default_control_d](#)
- `#define` [ma97_analyse](#) [ma97_analyse_d](#)
- `#define` [ma97_analyse_coord](#) [ma97_analyse_coord_d](#)
- `#define` [ma97_factor](#) [ma97_factor_d](#)
- `#define` [ma97_factor_solve](#) [ma97_factor_solve_d](#)

- `#define ma97_solve ma97_solve_d`
- `#define ma97_free_akeep ma97_free_akeep_d`
- `#define ma97_free_fkeep ma97_free_fkeep_d`
- `#define ma97_finalise ma97_finalise_d`
- `#define ma97_enquire_posdef ma97_enquire_posdef_d`
- `#define ma97_enquire_indef ma97_enquire_indef_d`
- `#define ma97_alter ma97_alter_d`
- `#define ma97_solve_fredholm ma97_solve_fredholm_d`
- `#define ma97_lmultiply ma97_lmultiply_d`
- `#define ma97_sparse_fwd_solve ma97_sparse_fwd_solve_d`

Typedefs

- `typedef double ma97pkgtype_d`
- `typedef double ma97realtype_d`

Functions

- `void ma97_default_control_d (struct ma97_control_d *control)`
- `void ma97_analyse_d (int check, int n, const int ptr[], const int row[], ma97pkgtype_d val[], void **akeep, const struct ma97_control_d *control, struct ma97_info_d *info, int order[])`
- `void ma97_analyse_coord_d (int n, int ne, const int row[], const int col[], ma97pkgtype_d val[], void **akeep, const struct ma97_control_d *control, struct ma97_info_d *info, int order[])`
- `void ma97_factor_d (int matrix_type, const int ptr[], const int row[], const ma97pkgtype_d val[], void **akeep, void **fkeep, const struct ma97_control_d *control, struct ma97_info_d *info, ma97realtype_d scale[])`
- `void ma97_factor_solve_d (int matrix_type, const int ptr[], const int row[], const ma97pkgtype_d val[], int nrhs, ma97pkgtype_d x[], int ldx, void **akeep, void **fkeep, const struct ma97_control_d *control, struct ma97_info_d *info, ma97realtype_d scale[])`
- `void ma97_solve_d (int job, int nrhs, ma97pkgtype_d x[], int ldx, void **akeep, void **fkeep, const struct ma97_control_d *control, struct ma97_info_d *info)`
- `void ma97_free_akeep_d (void **akeep)`
- `void ma97_free_fkeep_d (void **fkeep)`
- `void ma97_finalise_d (void **akeep, void **fkeep)`
- `void ma97_enquire_posdef_d (void **akeep, void **fkeep, const struct ma97_control *control, struct ma97_info *info, ma97realtype_d d[])`
- `void ma97_enquire_indef_d (void **akeep, void **fkeep, const struct ma97_control *control, struct ma97_info *info, int *piv_order, ma97pkgtype_d *d)`
- `void ma97_alter_d (const ma97pkgtype_d d[], void **akeep, void **fkeep, const struct ma97_control *control, struct ma97_info *info)`
- `void ma97_solve_fredholm_d (int nrhs, int flag_out[], ma97pkgtype_d x[], int ldx, void **akeep, void **fkeep, const struct ma97_control_d *control, struct ma97_info_d *info)`
- `void ma97_lmultiply_d (int trans, int k, const ma97pkgtype_d x[], int ldx, ma97pkgtype_d y[], int ldy, void **akeep, void **fkeep, const struct ma97_control_d *control, struct ma97_info_d *info)`
- `void ma97_sparse_fwd_solve_d (int nbi, const int bindex[], const ma97pkgtype_d b[], const int order[], int *nxi, int xindex[], ma97pkgtype_d x[], void **akeep, void **fkeep, const struct ma97_control_d *control, struct ma97_info_d *info)`

7.78.1 Macro Definition Documentation

7.78.1.1 `#define ma97_control ma97_control_d`

Definition at line 13 of file hsl_ma97d.h.

7.78.1.2 **#define ma97_info ma97_info_d**

Definition at line 14 of file hsl_ma97d.h.

7.78.1.3 **#define ma97_default_control ma97_default_control_d**

Definition at line 15 of file hsl_ma97d.h.

7.78.1.4 **#define ma97_analyse ma97_analyse_d**

Definition at line 16 of file hsl_ma97d.h.

7.78.1.5 **#define ma97_analyse_coord ma97_analyse_coord_d**

Definition at line 17 of file hsl_ma97d.h.

7.78.1.6 **#define ma97_factor ma97_factor_d**

Definition at line 18 of file hsl_ma97d.h.

7.78.1.7 **#define ma97_factor_solve ma97_factor_solve_d**

Definition at line 19 of file hsl_ma97d.h.

7.78.1.8 **#define ma97_solve ma97_solve_d**

Definition at line 20 of file hsl_ma97d.h.

7.78.1.9 **#define ma97_free_akeep ma97_free_akeep_d**

Definition at line 21 of file hsl_ma97d.h.

7.78.1.10 **#define ma97_free_fkeep ma97_free_fkeep_d**

Definition at line 22 of file hsl_ma97d.h.

7.78.1.11 **#define ma97_finalise ma97_finalise_d**

Definition at line 23 of file hsl_ma97d.h.

7.78.1.12 **#define ma97_enquire_posdef ma97_enquire_posdef_d**

Definition at line 24 of file hsl_ma97d.h.

7.78.1.13 **#define ma97_enquire_indef ma97_enquire_indef_d**

Definition at line 25 of file hsl_ma97d.h.

7.78.1.14 **#define ma97_alter ma97_alter_d**

Definition at line 26 of file hsl_ma97d.h.

7.78.1.15 **#define ma97_solve_fredholm ma97_solve_fredholm_d**

Definition at line 27 of file hsl_ma97d.h.

7.78.1.16 `#define ma97_multiply ma97_multiply_d`

Definition at line 28 of file `hsl_ma97d.h`.

7.78.1.17 `#define ma97_sparse_fwd_solve ma97_sparse_fwd_solve_d`

Definition at line 29 of file `hsl_ma97d.h`.

7.78.2 Typedef Documentation

7.78.2.1 `typedef double ma97pkgtype_d_`

Definition at line 32 of file `hsl_ma97d.h`.

7.78.2.2 `typedef double ma97realtype_d_`

Definition at line 33 of file `hsl_ma97d.h`.

7.78.3 Function Documentation

7.78.3.1 `void ma97_default_control_d (struct ma97_control_d * control)`

`void ma97_analyse_d (int check, int n, const int ptr[], const int row[], ma97pkgtype_d_val[], void ** akeep, const struct ma97_control_d * control, struct ma97_info_d * info, int order[])`

`void ma97_analyse_coord_d (int n, int ne, const int row[], const int col[], ma97pkgtype_d_val[], void ** akeep, const struct ma97_control_d * control, struct ma97_info_d * info, int order[])`

`void ma97_factor_d (int matrix_type, const int ptr[], const int row[], const ma97pkgtype_d_val[], void ** akeep, void ** fkeep, const struct ma97_control_d * control, struct ma97_info_d * info, ma97realtype_d_scale[])`

`void ma97_factor_solve_d (int matrix_type, const int ptr[], const int row[], const ma97pkgtype_d_val[], int nrhs, ma97pkgtype_d_x[], int ldx, void ** akeep, void ** fkeep, const struct ma97_control_d * control, struct ma97_info_d * info, ma97realtype_d_scale[])`

`void ma97_solve_d (int job, int nrhs, ma97pkgtype_d_x[], int ldx, void ** akeep, void ** fkeep, const struct ma97_control_d * control, struct ma97_info_d * info)`

`void ma97_free_akeep_d (void ** akeep)`

`void ma97_free_fkeep_d (void ** fkeep)`

`void ma97_finalise_d (void ** akeep, void ** fkeep)`

`void ma97_enquire_posdef_d (void ** akeep, void ** fkeep, const struct ma97_control * control, struct ma97_info * info, ma97realtype_d_d[])`

`void ma97_enquire_indef_d (void ** akeep, void ** fkeep, const struct ma97_control * control, struct ma97_info * info, int * piv_order, ma97pkgtype_d_d * d)`

`void ma97_alter_d (const ma97pkgtype_d_d[], void ** akeep, void ** fkeep, const struct ma97_control * control, struct ma97_info * info)`

- 7.78.3.13 void `ma97_solve_fredholm_d` (int *nrhs*, int *flag_out*[], *ma97pkgtype_d_x*[], int *ldx*, void ** *akeep*, void ** *fkeep*, const struct *ma97_control_d* * *control*, struct *ma97_info_d* * *info*)
- 7.78.3.14 void `ma97_lmultiply_d` (int *trans*, int *k*, const *ma97pkgtype_d_x*[], int *ldx*, *ma97pkgtype_d_y*[], int *ldy*, void ** *akeep*, void ** *fkeep*, const struct *ma97_control_d* * *control*, struct *ma97_info_d* * *info*)
- 7.78.3.15 void `ma97_sparse_fwd_solve_d` (int *nbi*, const int *bindex*[], const *ma97pkgtype_d_b*[], const int *order*[], int * *nxi*, int *xindex*[], *ma97pkgtype_d_x*[], void ** *akeep*, void ** *fkeep*, const struct *ma97_control_d* * *control*, struct *ma97_info_d* * *info*)

7.79 Algorithm/LinearSolvers/hsl_mc68i.h File Reference

```
#include "IpoptConfig.h"
#include "CoinHslConfig.h"
```

Classes

- struct [mc68_control](#)
- struct [mc68_info](#)

Macros

- #define [mc68_control](#) `mc68_control_i`
- #define [mc68_info](#) `mc68_info_i`
- #define [mc68_default_control](#) `mc68_default_control_i`
- #define [mc68_order](#) `mc68_order_i`

Functions

- void [mc68_default_control](#) (struct [mc68_control](#) **control*)
- void [mc68_order](#) (int *ord*, int *n*, const int *ptr*[], const int *row*[], int *perm*[], const struct [mc68_control](#) **control*, struct [mc68_info](#) **info*)

7.79.1 Macro Definition Documentation

7.79.1.1 #define [mc68_control](#) `mc68_control_i`

Definition at line 20 of file `hsl_mc68i.h`.

7.79.1.2 #define [mc68_info](#) `mc68_info_i`

Definition at line 21 of file `hsl_mc68i.h`.

7.79.1.3 #define [mc68_default_control](#) `mc68_default_control_i`

Definition at line 22 of file `hsl_mc68i.h`.

7.79.1.4 #define [mc68_order](#) `mc68_order_i`

Definition at line 23 of file `hsl_mc68i.h`.

7.79.2 Function Documentation

7.79.2.1 `void mc68_default_control (struct mc68_control * control)`

7.79.2.2 `void mc68_order (int ord, int n, const int ptr[], const int row[], int perm[], const struct mc68_control * control, struct mc68_info * info)`

7.80 Algorithm/LinearSolvers/lpGenKKTSolverInterface.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpAlgStrategy.hpp"
#include "IpSymLinearSolver.hpp"
```

Classes

- class [Ipopt::GenKKTSolverInterface](#)
Base class for interfaces to symmetric indefinite linear solvers for generic matrices.

Namespaces

- [Ipopt](#)

7.81 Algorithm/LinearSolvers/lpIterativeWsmSolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
```

Classes

- class [Ipopt::IterativeWsmSolverInterface](#)
Interface to the linear solver WISMP, derived from [SparseSymLinearSolverInterface](#).

Namespaces

- [Ipopt](#)

7.82 Algorithm/LinearSolvers/lpLinearSolversRegOp.hpp File Reference

```
#include "IpSmartPtr.hpp"
```

Namespaces

- [Ipopt](#)

Functions

- void [lpopt::RegisterOptions_LinearSolvers](#) (const SmartPtr< RegisteredOptions > &options)

7.83 Algorithm/LinearSolvers/lpMa27TSolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
```

Classes

- class [lpopt::Ma27TSolverInterface](#)
Interface to the symmetric linear solver MA27, derived from [SparseSymLinearSolverInterface](#).

Namespaces

- [lpopt](#)

7.84 Algorithm/LinearSolvers/lpMa28TDependencyDetector.hpp File Reference

```
#include "IpTDependencyDetector.hpp"
```

Classes

- class [lpopt::Ma28TDependencyDetector](#)
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

Namespaces

- [lpopt](#)

7.85 Algorithm/LinearSolvers/lpMa57TSolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
```

Classes

- class [lpopt::Ma57TSolverInterface](#)
Interface to the symmetric linear solver MA57, derived from [SparseSymLinearSolverInterface](#).

Namespaces

- [lpopt](#)

Typedefs

- typedef [ipfint ma57int](#)

7.85.1 Typedef Documentation

7.85.1.1 typedef [ipfint ma57int](#)

Definition at line 19 of file IpMa57TSolverInterface.hpp.

7.86 Algorithm/LinearSolvers/IpMa77SolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
#include "hsl_ma77d.h"
```

Classes

- class [Ipopt::Ma77SolverInterface](#)
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

Namespaces

- [Ipopt](#)

7.87 Algorithm/LinearSolvers/IpMa86SolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
#include "hsl_ma86d.h"
```

Classes

- class [Ipopt::Ma86SolverInterface](#)
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

Namespaces

- [Ipopt](#)

7.88 Algorithm/LinearSolvers/IpMa97SolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
#include "hsl_ma97d.h"
```


Classes

- class [Ipopt::Ma97SolverInterface](#)

Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

Namespaces

- [Ipopt](#)

7.89 Algorithm/LinearSolvers/IpMc19TSymScalingMethod.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpTSymScalingMethod.hpp"
```

Classes

- class [Ipopt::Mc19TSymScalingMethod](#)

Class for the method for computing scaling factors for symmetric matrices in triplet format, using MC19.

Namespaces

- [Ipopt](#)

7.90 Algorithm/LinearSolvers/IpMumpsSolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
```

Classes

- class [Ipopt::MumpsSolverInterface](#)

Interface to the linear solver Mumps, derived from [SparseSymLinearSolverInterface](#).

Namespaces

- [Ipopt](#)

7.91 Algorithm/LinearSolvers/IpPardisoSolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
```

Classes

- class [Ipopt::PardisoSolverInterface](#)

Interface to the linear solver Pardiso, derived from [SparseSymLinearSolverInterface](#).

Namespaces

- [Ipopt](#)

7.92 Algorithm/LinearSolvers/IpSlackBasedTSymScalingMethod.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpTSymScalingMethod.hpp"
```

Classes

- class [Ipopt::SlackBasedTSymScalingMethod](#)
Class for the method for computing scaling factors for symmetric matrices in triplet format, specifically for the inexact algorithm.

Namespaces

- [Ipopt](#)

7.93 Algorithm/LinearSolvers/IpSparseSymLinearSolverInterface.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpAlgStrategy.hpp"
#include "IpSymLinearSolver.hpp"
```

Classes

- class [Ipopt::SparseSymLinearSolverInterface](#)
Base class for interfaces to symmetric indefinite linear solvers for sparse matrices.

Namespaces

- [Ipopt](#)

7.94 Algorithm/LinearSolvers/IpSymLinearSolver.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
#include "IpAlgStrategy.hpp"
#include <vector>
```

Classes

- class [Ipopt::SymLinearSolver](#)
Base class for all derived symmetric linear solvers.

Namespaces

- [Ipopt](#)

Enumerations

- enum [Ipopt::ESymSolverStatus](#) {
[Ipopt::SYMSOLVER_SUCCESS](#), [Ipopt::SYMSOLVER_SINGULAR](#), [Ipopt::SYMSOLVER_WRONG_INERTIA](#),
[Ipopt::SYMSOLVER_CALL_AGAIN](#),
[Ipopt::SYMSOLVER_FATAL_ERROR](#) }

Enum to report outcome of a linear solve.

7.95 Algorithm/LinearSolvers/IpTDependencyDetector.hpp File Reference

```
#include "IpAlgStrategy.hpp"
#include <list>
```

Classes

- class [Ipopt::TDependencyDetector](#)
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

Namespaces

- [Ipopt](#)

7.96 Algorithm/LinearSolvers/IpTripletToCSRConverter.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpReferenced.hpp"
```

Classes

- class [Ipopt::TripletToCSRConverter](#)
Class for converting symmetric matrices given in triplet format to matrices in compressed sparse row (CSR) format of the upper triangular part (or, equivalently, compressed sparse column (CSC) format for the lower triangular part).
- class [Ipopt::TripletToCSRConverter::TripletEntry](#)
Class for one triplet position entry.

Namespaces

- [Ipopt](#)

7.97 Algorithm/LinearSolvers/IpTSymDependencyDetector.hpp File Reference

```
#include "IpTDependencyDetector.hpp"
#include "IpTSymLinearSolver.hpp"
```

Classes

- class [Ipopt::TSymDependencyDetector](#)
Base class for all derived algorithms for detecting linearly dependent rows in the constraint Jacobian.

Namespaces

- [Ipopt](#)

7.98 Algorithm/LinearSolvers/IpTSymLinearSolver.hpp File Reference

```
#include "IpSymLinearSolver.hpp"
#include "IpSparseSymLinearSolverInterface.hpp"
#include "IpTSymScalingMethod.hpp"
#include "IpSymMatrix.hpp"
#include "IpTripletToCSRConverter.hpp"
#include <vector>
#include <list>
```

Classes

- class [Ipopt::TSymLinearSolver](#)
General driver for linear solvers for sparse indefinite symmetric matrices.

Namespaces

- [Ipopt](#)

7.99 Algorithm/LinearSolvers/IpTSymScalingMethod.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpAlgStrategy.hpp"
```

Classes

- class [Ipopt::TSymScalingMethod](#)
Base class for the method for computing scaling factors for symmetric matrices in triplet format.

Namespaces

- [lpopt](#)

Functions

- [lpopt::DECLARE_STD_EXCEPTION](#) (ERROR_IN_LINEAR_SCALING_METHOD)

7.100 Algorithm/LinearSolvers/lpWsmSolverInterface.hpp File Reference

```
#include "IpSparseSymLinearSolverInterface.hpp"
```

Classes

- class [lpopt::WsmSolverInterface](#)
Interface to the linear solver Wsm, derived from [SparseSymLinearSolverInterface](#).

Namespaces

- [lpopt](#)

7.101 Apps/AmplSolver/AmplTNLP.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpTNLP.hpp"
#include "IpJournalist.hpp"
#include "IpOptionsList.hpp"
#include <map>
#include <string>
```

Classes

- class [lpopt::AmplSuffixHandler](#)
- class [lpopt::AmplOptionsList](#)
Class for storing a number of AMPL options that should be registered to the AMPL Solver library interface.
- class [lpopt::AmplOptionsList::AmplOption](#)
Ampl Option class, contains name, type and description for an AMPL option.
- class [lpopt::AmplOptionsList::PrivatInfo](#)
- class [lpopt::AmplTNLP](#)
Ampl Interface.

Namespaces

- [lpopt](#)

7.102 Common/config_default.h File Reference

```
#include "configall_system.h"
#include "config_ipopt_default.h"
```

Macros

- `#define COIN_IPOPT_CHECKLEVEL 0`
- `#define COIN_IPOPT_VERBOSITY 0`
- `#define COIN_HAS_AS_L 1`
- `#define COIN_HAS_BLAS 1`
- `#define COIN_HAS_LAPACK 1`
- `#define COIN_HAS_HSL 1`
- `#define FORTRAN_INTEGER_TYPE int`

7.102.1 Macro Definition Documentation

7.102.1.1 `#define COIN_IPOPT_CHECKLEVEL 0`

Definition at line 14 of file config_default.h.

7.102.1.2 `#define COIN_IPOPT_VERBOSITY 0`

Definition at line 17 of file config_default.h.

7.102.1.3 `#define COIN_HAS_AS_L 1`

Definition at line 20 of file config_default.h.

7.102.1.4 `#define COIN_HAS_BLAS 1`

Definition at line 23 of file config_default.h.

7.102.1.5 `#define COIN_HAS_LAPACK 1`

Definition at line 26 of file config_default.h.

7.102.1.6 `#define COIN_HAS_HSL 1`

Definition at line 29 of file config_default.h.

7.102.1.7 `#define FORTRAN_INTEGER_TYPE int`

Definition at line 58 of file config_default.h.

7.103 Common/config_ipopt_default.h File Reference

Macros

- `#define IPOPT_VERSION "trunk"`
- `#define IPOPT_VERSION_MAJOR 9999`
- `#define IPOPT_VERSION_MINOR 9999`
- `#define IPOPT_VERSION_RELEASE 9999`

7.103.1 Macro Definition Documentation

7.103.1.1 `#define IPOPT_VERSION "trunk"`

Definition at line 8 of file `config_ipopt_default.h`.

7.103.1.2 `#define IPOPT_VERSION_MAJOR 9999`

Definition at line 11 of file `config_ipopt_default.h`.

7.103.1.3 `#define IPOPT_VERSION_MINOR 9999`

Definition at line 14 of file `config_ipopt_default.h`.

7.103.1.4 `#define IPOPT_VERSION_RELEASE 9999`

Definition at line 17 of file `config_ipopt_default.h`.

7.104 Common/IpCachedResults.hpp File Reference

```
#include "IpTaggedObject.hpp"
#include "IpObserver.hpp"
#include <algorithm>
#include <vector>
#include <list>
```

Classes

- class [Ipopt::DependentResult< T >](#)
Templated class which stores one entry for the CachedResult class.
- class [Ipopt::CachedResults< T >](#)
Cache Priority Enum.
- class [Ipopt::DependentResult< T >](#)
Templated class which stores one entry for the CachedResult class.

Namespaces

- [Ipopt](#)

7.105 Common/IpDebug.hpp File Reference

```
#include "IpoptConfig.h"
#include "IpTypes.hpp"
```

Macros

- `#define COIN_IPOPT_CHECKLEVEL 0`
- `#define DBG_ASSERT(test)`

- `#define DBG_ASSERT_EXCEPTION(__condition, __except_type, __msg)`
- `#define DBG_DO(__cmd)`
- `#define COIN_IPOPT_VERBOSITY 0`
- `#define DBG_START_FUN(__func_name, __verbose_level)`
- `#define DBG_START_METH(__func_name, __verbose_level)`
- `#define DBG_PRINT(__printf_args)`
- `#define DBG_PRINT_VECTOR(__verbose_level, __vec_name, __vec)`
- `#define DBG_PRINT_MATRIX(__verbose_level, __mat_name, __mat)`
- `#define DBG_EXEC(__verbosity, __cmd)`
- `#define DBG_VERBOSITY() 0`

7.105.1 Macro Definition Documentation

7.105.1.1 `#define COIN_IPOPT_CHECKLEVEL 0`

Definition at line 26 of file IpDebug.hpp.

7.105.1.2 `#define DBG_ASSERT(test)`

Definition at line 38 of file IpDebug.hpp.

7.105.1.3 `#define DBG_ASSERT_EXCEPTION(__condition, __except_type, __msg)`

Definition at line 39 of file IpDebug.hpp.

7.105.1.4 `#define DBG_DO(__cmd)`

Definition at line 40 of file IpDebug.hpp.

7.105.1.5 `#define COIN_IPOPT_VERBOSITY 0`

Definition at line 44 of file IpDebug.hpp.

7.105.1.6 `#define DBG_START_FUN(__func_name, __verbose_level)`

Definition at line 48 of file IpDebug.hpp.

7.105.1.7 `#define DBG_START_METH(__func_name, __verbose_level)`

Definition at line 49 of file IpDebug.hpp.

7.105.1.8 `#define DBG_PRINT(__printf_args)`

Definition at line 50 of file IpDebug.hpp.

7.105.1.9 `#define DBG_PRINT_VECTOR(__verbose_level, __vec_name, __vec)`

Definition at line 51 of file IpDebug.hpp.

7.105.1.10 `#define DBG_PRINT_MATRIX(__verbose_level, __mat_name, __mat)`

Definition at line 52 of file IpDebug.hpp.

7.105.1.11 `#define DBG_EXEC(__verbosity, __cmd)`

Definition at line 53 of file IpDebug.hpp.

7.105.1.12 `#define DBG_VERBOSITY() 0`

Definition at line 54 of file IpDebug.hpp.

7.106 Common/IpException.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpJournalist.hpp"
```

Classes

- class [Ipopt::IpoptException](#)
This is the base class for all exceptions.

Namespaces

- [Ipopt](#)

Macros

- `#define THROW_EXCEPTION(__except_type, __msg) throw __except_type((__msg), (__FILE__), (__LINE__));`
- `#define ASSERT_EXCEPTION(__condition, __except_type, __msg)`
- `#define DECLARE_STD_EXCEPTION(__except_type)`

7.106.1 Macro Definition Documentation

7.106.1.1 `#define THROW_EXCEPTION(__except_type, __msg) throw __except_type((__msg), (__FILE__), (__LINE__));`

Definition at line 123 of file IpException.hpp.

7.106.1.2 `#define ASSERT_EXCEPTION(__condition, __except_type, __msg)`

Value:

```
if (! (__condition) ) { \
    std::string newmsg = #__condition; \
    newmsg += " evaluated false: "; \
    newmsg += __msg; \
    throw __except_type( (newmsg), (__FILE__), (__LINE__) ); \
}
```

Definition at line 126 of file IpException.hpp.

7.106.1.3 `#define DECLARE_STD_EXCEPTION(__except_type)`

Value:

```
class __except_type : public Ipopt::IpoptException \
{ \
    public: \
        __except_type(std::string msg, std::string fname, Ipopt::Index line) \
        : Ipopt::IpoptException(msg, fname, line, #__except_type) {} \
}
```

```

    __except_type(const __except_type& copy) \
: Ipopt::IpoptException(copy) {} \
private: \
    __except_type(); \
    void operator=(const __except_type&); \
}

```

Definition at line 134 of file IpException.hpp.

7.107 Common/IpJournalist.hpp File Reference

```

#include "IpoptConfig.h"
#include "IpTypes.hpp"
#include "IpReferenced.hpp"
#include "IpSmartPtr.hpp"
#include <cstdlib>
#include <cstdio>
#include <string>
#include <vector>
#include <ostream>

```

Classes

- class [Ipopt::Journalist](#)
Class responsible for all message output.
- class [Ipopt::Journal](#)
Journal class (part of the [Journalist](#) implementation.).
- class [Ipopt::FileJournal](#)
FileJournal class.
- class [Ipopt::StreamJournal](#)
StreamJournal class.

Namespaces

- [Ipopt](#)

Enumerations

Journalist Enumerations.

- enum [Ipopt::EJournalLevel](#) {
[Ipopt::J_INSUPPRESSIBLE](#) = -1, [Ipopt::J_NONE](#) = 0, [Ipopt::J_ERROR](#), [Ipopt::J_STRONGWARNING](#),
[Ipopt::J_SUMMARY](#), [Ipopt::J_WARNING](#), [Ipopt::J_ITERSUMMARY](#), [Ipopt::J_DETAILED](#),
[Ipopt::J_MOREDETAILED](#), [Ipopt::J_VECTOR](#), [Ipopt::J_MOREVECTOR](#), [Ipopt::J_MATRIX](#),
[Ipopt::J_MOREMATRIX](#), [Ipopt::J_ALL](#), [Ipopt::J_LAST_LEVEL](#) }
Print Level Enum.

- enum `Ipopt::EJournalCategory` {
`Ipopt::J_DBG = 0, Ipopt::J_STATISTICS, Ipopt::J_MAIN, Ipopt::J_INITIALIZATION,`
`Ipopt::J_BARRIER_UPDATE, Ipopt::J_SOLVE_PD_SYSTEM, Ipopt::J_FRAC_TO_BOUND, Ipopt::J_LINEAR_ALGEBRA,`
`Ipopt::J_LINE_SEARCH, Ipopt::J_HESSIAN_APPROXIMATION, Ipopt::J_SOLUTION, Ipopt::J_DOCUMENTATION,`
`Ipopt::J_NLP, Ipopt::J_TIMING_STATISTICS, Ipopt::J_USER_APPLICATION, Ipopt::J_USER1,`
`Ipopt::J_USER2, Ipopt::J_USER3, Ipopt::J_USER4, Ipopt::J_USER5,`
`Ipopt::J_USER6, Ipopt::J_USER7, Ipopt::J_USER8, Ipopt::J_USER9,`
`Ipopt::J_USER10, Ipopt::J_USER11, Ipopt::J_USER12, Ipopt::J_USER13,`
`Ipopt::J_USER14, Ipopt::J_USER15, Ipopt::J_USER16, Ipopt::J_USER17,`
`Ipopt::J_LAST_CATEGORY` }

Category Selection Enum.

7.108 Common/IpObserver.hpp File Reference

```
#include "IpUtils.hpp"
#include <vector>
#include <algorithm>
```

Classes

- class `Ipopt::Observer`
Slight Variation of the [Observer](#) Design Pattern.
- class `Ipopt::Subject`
Slight Variation of the [Observer](#) Design Pattern ([Subject](#) part).

Namespaces

- `Ipopt`

7.109 Common/IpoptConfig.h File Reference

```
#include "config_ipopt_default.h"
```

7.110 Common/IpOptionsList.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpReferenced.hpp"
#include "IpException.hpp"
#include "IpRegOptions.hpp"
#include <iostream>
#include <map>
```

Classes

- class [Ipopt::OptionsList](#)
This class stores a list of user set options.
- class [Ipopt::OptionsList::OptionValue](#)
Class for storing the value and counter for each option in [OptionsList](#).

Namespaces

- [Ipopt](#)

Functions

- [Ipopt::DECLARE_STD_EXCEPTION](#) (OPTION_INVALID)
Exception that can be used to indicate errors with options.

7.111 Common/IpReferenced.hpp File Reference

```
#include "IpTypes.hpp"
#include "IpDebug.hpp"
#include <list>
```

Classes

- class [Ipopt::Referencer](#)
Psydo-class, from which everything has to inherit that wants to use be registered as a [Referencer](#) for a [ReferencedObject](#).
- class [Ipopt::ReferencedObject](#)
[ReferencedObject](#) class.

Namespaces

- [Ipopt](#)

7.112 Common/IpRegOptions.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpReferenced.hpp"
#include "IpException.hpp"
#include "IpSmartPtr.hpp"
#include <map>
```

Classes

- class [Ipopt::RegisteredOption](#)
Base class for registered options.
- class [Ipopt::RegisteredOption::string_entry](#)
class to hold the valid string settings for a string option
- class [Ipopt::RegisteredOptions](#)
Class for storing registered options.

Namespaces

- [Ipopt](#)

Enumerations

- enum [Ipopt::RegisteredOptionType](#) { [Ipopt::OT_Number](#), [Ipopt::OT_Integer](#), [Ipopt::OT_String](#), [Ipopt::OT_Unknown](#) }

7.113 Common/IpSmartPtr.hpp File Reference

```
#include "IpReferenced.hpp"  
#include "IpDebug.hpp"
```

Classes

- class [Ipopt::SmartPtr< T >](#)
Template class for Smart Pointers.

Namespaces

- [Ipopt](#)

Macros

- `#define` [IPOPT_UNUSED](#)
- `#define` [ipopt_dbg_smartptr_verbosity](#) 0

Functions

- `template<class U1 , class U2 >`
`bool` [Ipopt::ComparePointers](#) (const U1 *lhs, const U2 *rhs)
- `template<class T >`
`void` [Ipopt::swap](#) (SmartPtr< T > &a, SmartPtr< T > &b)
- `template<class T >`
`bool` [Ipopt::operator<](#) (const SmartPtr< T > &lhs, const SmartPtr< T > &rhs)
- `template<class T >`
`bool` [Ipopt::operator>](#) (const SmartPtr< T > &lhs, const SmartPtr< T > &rhs)

- `template<class T >`
`bool lpopt::operator<= (const SmartPtr< T > &lhs, const SmartPtr< T > &rhs)`
- `template<class T >`
`bool lpopt::operator>= (const SmartPtr< T > &lhs, const SmartPtr< T > &rhs)`

SmartPtr friend function declarations.

- `template<class U >`
`U * lpopt::GetRawPtr (const SmartPtr< U > &smart_ptr)`
- `template<class U >`
`SmartPtr< const U > lpopt::ConstPtr (const SmartPtr< U > &smart_ptr)`
- `template<class U >`
`bool lpopt::IsNull (const SmartPtr< U > &smart_ptr)`
- `template<class U >`
`bool lpopt::IsValid (const SmartPtr< U > &smart_ptr)`
- `template<class U1 , class U2 >`
`bool lpopt::operator== (const SmartPtr< U1 > &lhs, const SmartPtr< U2 > &rhs)`
- `template<class U1 , class U2 >`
`bool lpopt::operator== (const SmartPtr< U1 > &lhs, U2 *raw_rhs)`
- `template<class U1 , class U2 >`
`bool lpopt::operator== (U1 *lhs, const SmartPtr< U2 > &raw_rhs)`
- `template<class U1 , class U2 >`
`bool lpopt::operator!= (const SmartPtr< U1 > &lhs, const SmartPtr< U2 > &rhs)`
- `template<class U1 , class U2 >`
`bool lpopt::operator!= (const SmartPtr< U1 > &lhs, U2 *raw_rhs)`
- `template<class U1 , class U2 >`
`bool lpopt::operator!= (U1 *lhs, const SmartPtr< U2 > &raw_rhs)`

7.113.1 Macro Definition Documentation

7.113.1.1 `#define IPOPT_UNUSED`

Definition at line 22 of file `IpSmartPtr.hpp`.

7.113.1.2 `#define ipopt_dbg_smartptr_verbosity 0`

Definition at line 175 of file `IpSmartPtr.hpp`.

7.114 Common/IpTaggedObject.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpDebug.hpp"
#include "IpReferenced.hpp"
#include "IpObserver.hpp"
#include <limits>
#include <utility>
```

Classes

- class [lpopt::TaggedObject](#)
TaggedObject class.

Namespaces

- [Ipopt](#)

Functions

- TaggedObject::Tag [Ipopt::operator+](#) (const TaggedObject::Tag &tag1, const TaggedObject::Tag &tag2)
The addition of two tags - do not use.

7.115 Common/IpTimedTask.hpp File Reference

```
#include "IpUtils.hpp"
```

Classes

- class [Ipopt::TimedTask](#)
This class is used to collect timing information for a particular task.

Namespaces

- [Ipopt](#)

7.116 Common/IpTypes.hpp File Reference

```
#include "IpoptConfig.h"
```

Namespaces

- [Ipopt](#)

Typedefs

- typedef double [Ipopt::Number](#)
Type of all numbers.
- typedef int [Ipopt::Index](#)
Type of all indices of vectors, matrices etc.
- typedef int [Ipopt::Int](#)
Type of default integer.
- typedef [FORTRAN_INTEGER_TYPE](#) ipfint

7.116.1 Typedef Documentation

7.116.1.1 typedef FORTRAN_INTEGER_TYPE ipfint

Definition at line 26 of file IpTypes.hpp.

7.117 Common/IpUtils.hpp File Reference

```
#include "IpTypes.hpp"
#include "IpDebug.hpp"
```

Namespaces

- [Ipopt](#)

Functions

- [Index Ipopt::Max](#) ([Index](#) a, [Index](#) b)
- [Index Ipopt::Max](#) ([Index](#) a, [Index](#) b, [Index](#) c)
- [Index Ipopt::Max](#) ([Index](#) a, [Index](#) b, [Index](#) c, [Index](#) d)
- [Index Ipopt::Min](#) ([Index](#) a, [Index](#) b)
- [Index Ipopt::Min](#) ([Index](#) a, [Index](#) b, [Index](#) c)
- [Index Ipopt::Min](#) ([Index](#) a, [Index](#) b, [Index](#) c, [Index](#) d)
- [Number Ipopt::Max](#) ([Number](#) a, [Number](#) b)
- [Number Ipopt::Max](#) ([Number](#) a, [Number](#) b, [Number](#) c)
- [Number Ipopt::Max](#) ([Number](#) a, [Number](#) b, [Number](#) c, [Number](#) d)
- [Number Ipopt::Min](#) ([Number](#) a, [Number](#) b)
- [Number Ipopt::Min](#) ([Number](#) a, [Number](#) b, [Number](#) c)
- [Number Ipopt::Min](#) ([Number](#) a, [Number](#) b, [Number](#) c, [Number](#) d)
- [bool Ipopt::IsFiniteNumber](#) ([Number](#) val)
Function returning true iff the argument is a valid double number (not NaN or Inf).
- [Number Ipopt::IpRandom01](#) ()
Function returning a random number between 0 and 1.
- [void Ipopt::IpResetRandom01](#) ()
Function resetting the random number generator.
- [Number Ipopt::CpuTime](#) ()
method determining CPU time
- [Number Ipopt::SysTime](#) ()
method determining system time
- [Number Ipopt::WallclockTime](#) ()
method determining wallclock time since first call
- [bool Ipopt::Compare_Le](#) ([Number](#) lhs, [Number](#) rhs, [Number](#) BasVal)
Method for comparing two numbers within machine precision.
- [int Ipopt::Snprintf](#) (char *str, long size, const char *format,...)
Method for printing a formatted output to a string with given size.

7.118 contrib/CGPenalty/IpCGPenaltyCq.hpp File Reference

```
#include "IpIpoptCalculatedQuantities.hpp"
#include "IpCGPenaltyData.hpp"
```


Classes

- class [lpopt::CGPenaltyCq](#)
Class for all Chen-Goldfarb penalty method specific calculated quantities.

Namespaces

- [lpopt](#)

7.119 contrib/CGPenalty/lpCGPenaltyData.hpp File Reference

```
#include "IpIteratesVector.hpp"  
#include "IpOptionsList.hpp"  
#include "IpIpoptData.hpp"
```

Classes

- class [lpopt::CGPenaltyData](#)
Class to organize all the additional data required by the Chen-Goldfarb penalty function algorithm.

Namespaces

- [lpopt](#)

7.120 contrib/CGPenalty/lpCGPenaltyLSAccepter.hpp File Reference

```
#include "IpPiecewisePenalty.hpp"  
#include "IpBacktrackingLSAccepter.hpp"  
#include "IpPDSystemSolver.hpp"  
#include "IpIpoptAlg.hpp"  
#include "IpCGPenaltyCq.hpp"
```

Classes

- class [lpopt::CGPenaltyLSAccepter](#)
Line search acceptor, based on the Chen-Goldfarb penalty function approach.

Namespaces

- [lpopt](#)

7.121 contrib/CGPenalty/lpCGPenaltyRegOp.hpp File Reference

```
#include "IpSmartPtr.hpp"
```

Namespaces

- [lpopt](#)

Functions

- void [lpopt::RegisterOptions_CGPenalty](#) (const SmartPtr< RegisteredOptions > &options)

7.122 contrib/CGPenalty/lpCGPerturbationHandler.hpp File Reference

```
#include "IpPDPerturbationHandler.hpp"
#include "IpCGPenaltyCq.hpp"
```

Classes

- class [lpopt::CGPerturbationHandler](#)

Class for handling the perturbation factors δ_x , δ_s , δ_c , and δ_d in the primal dual system.

Namespaces

- [lpopt](#)

7.123 contrib/CGPenalty/lpCGSearchDirCalc.hpp File Reference

```
#include "IpSearchDirCalculator.hpp"
#include "IpPDSystemSolver.hpp"
#include "IpCGPenaltyCq.hpp"
```

Classes

- class [lpopt::CGSearchDirCalculator](#)

Implementation of the search direction calculator that computes the Chen-Goldfarb step for the current barrier and penalty parameter.

Namespaces

- [lpopt](#)

7.124 contrib/CGPenalty/lpPiecewisePenalty.hpp File Reference

```
#include "IpJournalist.hpp"
#include "IpDebug.hpp"
#include "IpOptionsList.hpp"
#include "IpIoptCalculatedQuantities.hpp"
#include "IpBacktrackingLSAcceptor.hpp"
#include "IpPDSystemSolver.hpp"
#include <list>
#include <vector>
```

Classes

- struct [Ipopt::PiecewisePenEntry](#)
struct for one Piecewise Penalty entry.
- class [Ipopt::PiecewisePenalty](#)
Class for the Piecewise Penalty.

Namespaces

- [Ipopt](#)

Typedefs

- typedef struct
[Ipopt::PiecewisePenEntry](#) [Ipopt::PiecewisePenEntry](#)
struct for one Piecewise Penalty entry.

7.125 contrib/LinearSolverLoader/HSLLoader.h File Reference

```
#include "IpoptConfig.h"
```

Macros

- #define [ma77_control](#) [ma77_control_d](#)
- #define [ma77_info](#) [ma77_info_d](#)
- #define [ma77_default_control](#) [ma77_default_control_d](#)
- #define [ma77_open_nelt](#) [ma77_open_nelt_d](#)
- #define [ma77_open](#) [ma77_open_d](#)
- #define [ma77_input_vars](#) [ma77_input_vars_d](#)
- #define [ma77_input_reals](#) [ma77_input_reals_d](#)
- #define [ma77_analyse](#) [ma77_analyse_d](#)
- #define [ma77_factor](#) [ma77_factor_d](#)
- #define [ma77_factor_solve](#) [ma77_factor_solve_d](#)
- #define [ma77_solve](#) [ma77_solve_d](#)
- #define [ma77_resid](#) [ma77_resid_d](#)
- #define [ma77_scale](#) [ma77_scale_d](#)

- `#define ma77_enquire_posdef ma77_enquire_posdef_d`
- `#define ma77_enquire_indef ma77_enquire_indef_d`
- `#define ma77_alter ma77_alter_d`
- `#define ma77_restart ma77_restart_d`
- `#define ma77_finalise ma77_finalise_d`
- `#define ma86_control ma86_control_d`
- `#define ma86_info ma86_info_d`
- `#define ma86_default_control ma86_default_control_d`
- `#define ma86_analyse ma86_analyse_d`
- `#define ma86_factor ma86_factor_d`
- `#define ma86_factor_solve ma86_factor_solve_d`
- `#define ma86_solve ma86_solve_d`
- `#define ma86_finalise ma86_finalise_d`
- `#define ma97_control ma97_control_d`
- `#define ma97_info ma97_info_d`
- `#define ma97_default_control ma97_default_control_d`
- `#define ma97_analyse ma97_analyse_d`
- `#define ma97_factor ma97_factor_d`
- `#define ma97_factor_solve ma97_factor_solve_d`
- `#define ma97_solve ma97_solve_d`
- `#define ma97_finalise ma97_finalise_d`
- `#define ma97_free_akeep ma97_free_akeep_d`

Typedefs

- `typedef double ma77pkgtype_d_`
- `typedef double ma86pkgtype_d_`
- `typedef double ma86realtype_d_`
- `typedef double ma97pkgtype_d_`
- `typedef double ma97realtype_d_`
- `typedef FORTRAN_INTEGER_TYPE ipfint`
- `typedef void(* ma27ad_t)(ipfint *N, ipfint *NZ, const ipfint *IRN, const ipfint *ICN, ipfint *IW, ipfint *LIW, ipfint *IKEEP, ipfint *IW1, ipfint *NSTEPS, ipfint *IFLAG, ipfint *ICNTL, double *CNTL, ipfint *INFO, double *OPS)`
- `typedef void(* ma27bd_t)(ipfint *N, ipfint *NZ, const ipfint *IRN, const ipfint *ICN, double *A, ipfint *LA, ipfint *IW, ipfint *LIW, ipfint *IKEEP, ipfint *NSTEPS, ipfint *MAXFRT, ipfint *IW1, ipfint *ICNTL, double *CNTL, ipfint *INFO)`
- `typedef void(* ma27cd_t)(ipfint *N, double *A, ipfint *LA, ipfint *IW, ipfint *LIW, double *W, ipfint *MAXFRT, double *RHS, ipfint *IW1, ipfint *NSTEPS, ipfint *ICNTL, double *CNTL)`
- `typedef void(* ma27id_t)(ipfint *ICNTL, double *CNTL)`
- `typedef void(* ma28ad_t)(void *nsize, void *nz, void *rw, void *licn, void *iw, void *lirn, void *iw2, void *pivtol, void *iw3, void *iw4, void *rw2, void *iflag)`
- `typedef void(* ma57ad_t)(ipfint *n, ipfint *ne, const ipfint *irn, const ipfint *jcn, ipfint *lkeep, ipfint *keep, ipfint *iwork, ipfint *icntl, ipfint *info, double *rinfo)`
- `typedef void(* ma57bd_t)(ipfint *n, ipfint *ne, double *a, double *fact, ipfint *lfact, ipfint *ifact, ipfint *lifact, ipfint *lkeep, ipfint *keep, ipfint *iwork, ipfint *icntl, double *cntl, ipfint *info, double *rinfo)`
- `typedef void(* ma57cd_t)(ipfint *job, ipfint *n, double *fact, ipfint *lfact, ipfint *ifact, ipfint *lifact, ipfint *nrhs, double *rhs, ipfint *lrhs, double *work, ipfint *lwork, ipfint *iwork, ipfint *icntl, ipfint *info)`
- `typedef void(* ma57ed_t)(ipfint *n, ipfint *ic, ipfint *keep, double *fact, ipfint *lfact, double *newfac, ipfint *lnew, ipfint *ifact, ipfint *lifact, ipfint *newwifc, ipfint *linew, ipfint *info)`
- `typedef void(* ma57id_t)(double *cntl, ipfint *icntl)`
- `typedef void(* ma77_default_control_t)(struct ma77_control_d *control)`

- typedef void(* [ma77_open_nelt_t](#))(const int n, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info, const int nelt)
- typedef void(* [ma77_open_t](#))(const int n, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma77_input_vars_t](#))(const int idx, const int nvar, const int list[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma77_input_reals_t](#))(const int idx, const int length, const double reals[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma77_analyse_t](#))(const int order[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma77_factor_t](#))(const int posdef, void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info, const double *scale)
- typedef void(* [ma77_factor_solve_t](#))(const int posdef, void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info, const double *scale, const int nrhs, const int lx, double rhs[])
- typedef void(* [ma77_solve_t](#))(const int job, const int nrhs, const int lx, double x[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info, const double *scale)
- typedef void(* [ma77_resid_t](#))(const int nrhs, const int lx, const double x[], const int lresid, double resid[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info, double *anorm_bnd)
- typedef void(* [ma77_scale_t](#))(double scale[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info, double *anorm)
- typedef void(* [ma77_enquire_posdef_t](#))(double d[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma77_enquire_indef_t](#))(int piv_order[], double d[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma77_alter_t](#))(const double d[], void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma77_restart_t](#))(const char *restart_file, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma77_finalise_t](#))(void **keep, const struct [ma77_control_d](#) *control, struct [ma77_info_d](#) *info)
- typedef void(* [ma86_default_control_t](#))(struct [ma86_control](#) *control)
- typedef void(* [ma86_analyse_t](#))(const int n, const int ptr[], const int row[], int order[], void **keep, const struct [ma86_control](#) *control, struct [ma86_info](#) *info)
- typedef void(* [ma86_factor_t](#))(const int n, const int ptr[], const int row[], const [ma86pkgtype_d_val](#)[], const int order[], void **keep, const struct [ma86_control](#) *control, struct [ma86_info](#) *info, const [ma86pkgtype_d](#) scale[])
- typedef void(* [ma86_factor_solve_t](#))(const int n, const int ptr[], const int row[], const [ma86pkgtype_d_val](#)[], const int order[], void **keep, const struct [ma86_control](#) *control, struct [ma86_info](#) *info, const int nrhs, const int ldx, [ma86pkgtype_d_x](#)[], const [ma86pkgtype_d](#) scale[])
- typedef void(* [ma86_solve_t](#))(const int job, const int nrhs, const int ldx, [ma86pkgtype_d_x](#), const int order[], void **keep, const struct [ma86_control](#) *control, struct [ma86_info](#) *info, const [ma86pkgtype_d](#) scale[])
- typedef void(* [ma86_finalise_t](#))(void **keep, const struct [ma86_control](#) *control)
- typedef void(* [ma97_default_control_t](#))(struct [ma97_control](#) *control)
- typedef void(* [ma97_analyse_t](#))(const int check, const int n, const int ptr[], const int row[], [ma97pkgtype_d_val](#)[], void **akeep, const struct [ma97_control](#) *control, struct [ma97_info](#) *info, int order[])
- typedef void(* [ma97_factor_t](#))(const int matrix_type, const int ptr[], const int row[], const [ma97pkgtype_d_val](#)[], void **akeep, void **fkeep, const struct [ma97_control](#) *control, struct [ma97_info](#) *info, const [ma97pkgtype_d](#) scale[])
- typedef void(* [ma97_factor_solve_t](#))(const int matrix_type, const int ptr[], const int row[], const [ma97pkgtype_d_val](#)[], const int nrhs, [ma97pkgtype_d_x](#)[], const int ldx, void **akeep, void **fkeep, const struct [ma97_control](#) *control, struct [ma97_info](#) *info, const [ma97pkgtype_d](#) scale[])
- typedef void(* [ma97_solve_t](#))(const int job, const int nrhs, [ma97pkgtype_d_x](#), const int ldx, void **akeep, void **fkeep, const struct [ma97_control](#) *control, struct [ma97_info](#) *info)
- typedef void(* [ma97_finalise_t](#))(void **akeep, void **fkeep)

- typedef void(* [ma97_free_akeep_t](#))(void **akeep)
- typedef void(* [mc19ad_t](#))(ipfint *N, ipfint *NZ, double *A, ipfint *IRN, ipfint *ICN, float *R, float *C, float *W)
- typedef void(* [mc68_default_control_t](#))(struct mc68_control_i *control)
- typedef void(* [mc68_order_t](#))(int ord, int n, const int ptr[], const int row[], int perm[], const struct mc68_control_i *control, struct mc68_info_i *info)

Functions

- int [LSL_loadHSL](#) (const char *libname, char *msgbuf, int msglen)
Tries to load a dynamically linked library with HSL routines.
- int [LSL_unloadHSL](#) ()
Unloads a loaded HSL library.
- int [LSL_isHSLLoaded](#) ()
Indicates whether a HSL library has been loaded.
- int [LSL_isMA27available](#) ()
Indicates whether a HSL library is loaded and all symbols necessary to use MA27 have been found.
- int [LSL_isMA28available](#) ()
Indicates whether a HSL library is loaded and all symbols necessary to use MA28 have been found.
- int [LSL_isMA57available](#) ()
Indicates whether a HSL library is loaded and all symbols necessary to use MA57 have been found.
- int [LSL_isMA77available](#) ()
Indicates whether a HSL library is loaded and all symbols necessary to use MA77 have been found.
- int [LSL_isMA86available](#) ()
Indicates whether a HSL library is loaded and all symbols necessary to use HSL_MA86 have been found.
- int [LSL_isMA97available](#) ()
Indicates whether a HSL library is loaded and all symbols necessary to use HSL_MA97 have been found.
- int [LSL_isMC19available](#) ()
Indicates whether a HSL library is loaded and all symbols necessary to use MA57 have been found.
- int [LSL_isMC68available](#) ()
Indicates whether a HSL library is loaded and all symbols necessary to use HSL_MC68 have been found.
- char * [LSL_HSLLibraryName](#) ()
Returns name of the shared library that should contain HSL.
- void [LSL_setMA27](#) ([ma27ad_t](#) ma27ad, [ma27bd_t](#) ma27bd, [ma27cd_t](#) ma27cd, [ma27id_t](#) ma27id)
sets pointers to MA27 functions
- void [LSL_setMA28](#) ([ma28ad_t](#) ma28ad)
sets pointers to MA28 functions
- void [LSL_setMA57](#) ([ma57ad_t](#) ma57ad, [ma57bd_t](#) ma57bd, [ma57cd_t](#) ma57cd, [ma57ed_t](#) ma57ed, [ma57id_t](#) ma57id)
sets pointers to MA57 functions
- void [LSL_setMA77](#) ([ma77_default_control_t](#) ma77_default_control, [ma77_open_nelt_t](#) ma77_open_nelt, [ma77_open_t](#) ma77_open, [ma77_input_vars_t](#) ma77_input_vars, [ma77_input_reals_t](#) ma77_input_reals, [ma77_analyse_t](#) ma77_analyse, [ma77_factor_t](#) ma77_factor, [ma77_factor_solve_t](#) ma77_factor_solve, [ma77_solve_t](#) ma77_solve, [ma77_resid_t](#) ma77_resid, [ma77_scale_t](#) ma77_scale, [ma77_enquire_posdef_t](#) ma77_enquire_posdef, [ma77_enquire_indef_t](#) ma77_enquire_indef, [ma77_alter_t](#) ma77_alter, [ma77_restart_t](#) ma77_restart, [ma77_finalise_t](#) ma77_finalise)
sets pointers to MA77 functions
- void [LSL_setMA86](#) ([ma86_default_control_t](#) ma86_default_control, [ma86_analyse_t](#) ma86_analyse, [ma86_factor_t](#) ma86_factor, [ma86_factor_solve_t](#) ma86_factor_solve, [ma86_solve_t](#) ma86_solve, [ma86_finalise_t](#) ma86_finalise)

sets pointers to MA86 functions

- void [LSL_setMA97](#) ([ma97_default_control_t](#) [ma97_default_control](#), [ma97_analyse_t](#) [ma97_analyse](#), [ma97_factor_t](#) [ma97_factor](#), [ma97_factor_solve_t](#) [ma97_factor_solve](#), [ma97_solve_t](#) [ma97_solve](#), [ma97_finalise_t](#) [ma97_finalise](#), [ma97_free_akeep_t](#) [ma97_free_akeep](#))

sets pointers to MA97 functions

- void [LSL_setMC19](#) ([mc19ad_t](#) [mc19ad](#))

sets pointer to MC19 function

- void [LSL_setMC68](#) ([mc68_default_control_t](#) [mc68_default_control](#), [mc68_order_t](#) [mc68_order](#))

sets pointers to MC68 functions

7.125.1 Macro Definition Documentation

7.125.1.1 `#define ma77_control ma77_control_d`

Definition at line 20 of file HSLLoader.h.

7.125.1.2 `#define ma77_info ma77_info_d`

Definition at line 21 of file HSLLoader.h.

7.125.1.3 `#define ma77_default_control ma77_default_control_d`

Definition at line 22 of file HSLLoader.h.

7.125.1.4 `#define ma77_open_nelt ma77_open_nelt_d`

Definition at line 23 of file HSLLoader.h.

7.125.1.5 `#define ma77_open ma77_open_d`

Definition at line 24 of file HSLLoader.h.

7.125.1.6 `#define ma77_input_vars ma77_input_vars_d`

Definition at line 25 of file HSLLoader.h.

7.125.1.7 `#define ma77_input_reals ma77_input_reals_d`

Definition at line 26 of file HSLLoader.h.

7.125.1.8 `#define ma77_analyse ma77_analyse_d`

Definition at line 27 of file HSLLoader.h.

7.125.1.9 `#define ma77_factor ma77_factor_d`

Definition at line 28 of file HSLLoader.h.

7.125.1.10 `#define ma77_factor_solve ma77_factor_solve_d`

Definition at line 29 of file HSLLoader.h.

7.125.1.11 `#define ma77_solve ma77_solve_d`

Definition at line 30 of file HSLLoader.h.

7.125.1.12 **#define ma77_resid ma77_resid_d**

Definition at line 31 of file HSLLoader.h.

7.125.1.13 **#define ma77_scale ma77_scale_d**

Definition at line 32 of file HSLLoader.h.

7.125.1.14 **#define ma77_enquire_posdef ma77_enquire_posdef_d**

Definition at line 33 of file HSLLoader.h.

7.125.1.15 **#define ma77_enquire_indef ma77_enquire_indef_d**

Definition at line 34 of file HSLLoader.h.

7.125.1.16 **#define ma77_alter ma77_alter_d**

Definition at line 35 of file HSLLoader.h.

7.125.1.17 **#define ma77_restart ma77_restart_d**

Definition at line 36 of file HSLLoader.h.

7.125.1.18 **#define ma77_finalise ma77_finalise_d**

Definition at line 37 of file HSLLoader.h.

7.125.1.19 **#define ma86_control ma86_control_d**

Definition at line 46 of file HSLLoader.h.

7.125.1.20 **#define ma86_info ma86_info_d**

Definition at line 47 of file HSLLoader.h.

7.125.1.21 **#define ma86_default_control ma86_default_control_d**

Definition at line 48 of file HSLLoader.h.

7.125.1.22 **#define ma86_analyse ma86_analyse_d**

Definition at line 49 of file HSLLoader.h.

7.125.1.23 **#define ma86_factor ma86_factor_d**

Definition at line 50 of file HSLLoader.h.

7.125.1.24 **#define ma86_factor_solve ma86_factor_solve_d**

Definition at line 51 of file HSLLoader.h.

7.125.1.25 **#define ma86_solve ma86_solve_d**

Definition at line 52 of file HSLLoader.h.

7.125.1.26 #define ma86_finalise ma86_finalise_d

Definition at line 53 of file HSLLoader.h.

7.125.1.27 #define ma97_control ma97_control_d

Definition at line 62 of file HSLLoader.h.

7.125.1.28 #define ma97_info ma97_info_d

Definition at line 63 of file HSLLoader.h.

7.125.1.29 #define ma97_default_control ma97_default_control_d

Definition at line 64 of file HSLLoader.h.

7.125.1.30 #define ma97_analyse ma97_analyse_d

Definition at line 65 of file HSLLoader.h.

7.125.1.31 #define ma97_factor ma97_factor_d

Definition at line 66 of file HSLLoader.h.

7.125.1.32 #define ma97_factor_solve ma97_factor_solve_d

Definition at line 67 of file HSLLoader.h.

7.125.1.33 #define ma97_solve ma97_solve_d

Definition at line 68 of file HSLLoader.h.

7.125.1.34 #define ma97_finalise ma97_finalise_d

Definition at line 69 of file HSLLoader.h.

7.125.1.35 #define ma97_free_akeep ma97_free_akeep_d

Definition at line 70 of file HSLLoader.h.

7.125.2 Typedef Documentation**7.125.2.1 typedef double ma77pkgtype_d_**

Definition at line 41 of file HSLLoader.h.

7.125.2.2 typedef double ma86pkgtype_d_

Definition at line 57 of file HSLLoader.h.

7.125.2.3 typedef double ma86realtype_d_

Definition at line 59 of file HSLLoader.h.

7.125.2.4 typedef double ma97pkgtype_d_

Definition at line 74 of file HSLLoader.h.

7.125.2.5 `typedef double ma97realtype_d_`

Definition at line 76 of file HSLLoader.h.

7.125.2.6 `typedef FORTRAN_INTEGER_TYPE ipfint`

Definition at line 79 of file HSLLoader.h.

7.125.2.7 `typedef void(* ma27ad_t)(ipfint *N, ipfint *NZ, const ipfint *IRN, const ipfint *ICN, ipfint *IW, ipfint *LIW, ipfint *IKEEP, ipfint *IW1, ipfint *NSTEPS, ipfint *IFLAG, ipfint *ICNTL, double *CNTL, ipfint *INFO, double *OPS)`

Definition at line 86 of file HSLLoader.h.

7.125.2.8 `typedef void(* ma27bd_t)(ipfint *N, ipfint *NZ, const ipfint *IRN, const ipfint *ICN, double *A, ipfint *LA, ipfint *IW, ipfint *LIW, ipfint *IKEEP, ipfint *NSTEPS, ipfint *MAXFRT, ipfint *IW1, ipfint *ICNTL, double *CNTL, ipfint *INFO)`

Definition at line 90 of file HSLLoader.h.

7.125.2.9 `typedef void(* ma27cd_t)(ipfint *N, double *A, ipfint *LA, ipfint *IW, ipfint *LIW, double *W, ipfint *MAXFRT, double *RHS, ipfint *IW1, ipfint *NSTEPS, ipfint *ICNTL, double *CNTL)`

Definition at line 95 of file HSLLoader.h.

7.125.2.10 `typedef void(* ma27id_t)(ipfint *ICNTL, double *CNTL)`

Definition at line 99 of file HSLLoader.h.

7.125.2.11 `typedef void(* ma28ad_t)(void *nsize, void *nz, void *rw, void *licn, void *iw, void *lirn, void *iw2, void *pivtol, void *iw3, void *iw4, void *rw2, void *iflag)`

Definition at line 101 of file HSLLoader.h.

7.125.2.12 `typedef void(* ma57ad_t)(ipfint *n, ipfint *ne, const ipfint *irn, const ipfint *jcn, ipfint *lkeep, ipfint *keep, ipfint *iwork, ipfint *icntl, ipfint *info, double *rinfo)`

Definition at line 104 of file HSLLoader.h.

7.125.2.13 `typedef void(* ma57bd_t)(ipfint *n, ipfint *ne, double *a, double *fact, ipfint *lfact, ipfint *ifact, ipfint *lifact, ipfint *lkeep, ipfint *keep, ipfint *iwork, ipfint *icntl, double *cntl, ipfint *info, double *rinfo)`

Definition at line 117 of file HSLLoader.h.

7.125.2.14 `typedef void(* ma57cd_t)(ipfint *job, ipfint *n, double *fact, ipfint *lfact, ipfint *ifact, ipfint *lifact, ipfint *nrhs, double *rhs, ipfint *lrhs, double *work, ipfint *lwork, ipfint *iwork, ipfint *icntl, ipfint *info)`

Definition at line 133 of file HSLLoader.h.

7.125.2.15 `typedef void(* ma57ed_t)(ipfint *n, ipfint *ic, ipfint *keep, double *fact, ipfint *lfact, double *newfac, ipfint *lnew, ipfint *ifact, ipfint *lifact, ipfint *newwfc, ipfint *linew, ipfint *info)`

Definition at line 149 of file HSLLoader.h.

7.125.2.16 `typedef void(* ma57id_t)(double *cntl, ipfint *icntl)`

Definition at line 163 of file HSLLoader.h.

7.125.2.17 `typedef void(* ma77_default_control_t)(struct ma77_control_d *control)`

Definition at line 165 of file HSLLoader.h.

7.125.2.18 `typedef void(* ma77_open_nelt_t)(const int n, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, const int nelt)`

Definition at line 166 of file HSLLoader.h.

7.125.2.19 `typedef void(* ma77_open_t)(const int n, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 170 of file HSLLoader.h.

7.125.2.20 `typedef void(* ma77_input_vars_t)(const int idx, const int nvar, const int list[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 173 of file HSLLoader.h.

7.125.2.21 `typedef void(* ma77_input_reals_t)(const int idx, const int length, const double reals[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 175 of file HSLLoader.h.

7.125.2.22 `typedef void(* ma77_analyse_t)(const int order[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 178 of file HSLLoader.h.

7.125.2.23 `typedef void(* ma77_factor_t)(const int posdef, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, const double *scale)`

Definition at line 180 of file HSLLoader.h.

7.125.2.24 `typedef void(* ma77_factor_solve_t)(const int posdef, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, const double *scale, const int nrhs, const int lx, double rhs[])`

Definition at line 183 of file HSLLoader.h.

7.125.2.25 `typedef void(* ma77_solve_t)(const int job, const int nrhs, const int lx, double x[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, const double *scale)`

Definition at line 187 of file HSLLoader.h.

7.125.2.26 `typedef void(* ma77_resid_t)(const int nrhs, const int lx, const double x[], const int lresid, double resid[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, double *anorm_bnd)`

Definition at line 190 of file HSLLoader.h.

7.125.2.27 `typedef void(* ma77_scale_t)(double scale[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info, double *anorm)`

Definition at line 194 of file HSLLoader.h.

7.125.2.28 `typedef void(* ma77_enquire_posdef_t)(double d[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 197 of file HSLLoader.h.

7.125.2.29 `typedef void(* ma77_enquire_indef_t)(int piv_order[], double d[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 199 of file HSLLoader.h.

7.125.2.30 `typedef void(* ma77_alter_t)(const double d[], void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 201 of file HSLLoader.h.

7.125.2.31 `typedef void(* ma77_restart_t)(const char *restart_file, const char *fname1, const char *fname2, const char *fname3, const char *fname4, void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 203 of file HSLLoader.h.

7.125.2.32 `typedef void(* ma77_finalise_t)(void **keep, const struct ma77_control_d *control, struct ma77_info_d *info)`

Definition at line 206 of file HSLLoader.h.

7.125.2.33 `typedef void(* ma86_default_control_t)(struct ma86_control *control)`

Definition at line 209 of file HSLLoader.h.

7.125.2.34 `typedef void(* ma86_analyse_t)(const int n, const int ptr[], const int row[], int order[], void **keep, const struct ma86_control *control, struct ma86_info *info)`

Definition at line 210 of file HSLLoader.h.

7.125.2.35 `typedef void(* ma86_factor_t)(const int n, const int ptr[], const int row[], const ma86pkgtype_d_val[], const int order[], void **keep, const struct ma86_control *control, struct ma86_info *info, const ma86pkgtype_d_scale[])`

Definition at line 213 of file HSLLoader.h.

7.125.2.36 `typedef void(* ma86_factor_solve_t)(const int n, const int ptr[], const int row[], const ma86pkgtype_d_val[], const int order[], void **keep, const struct ma86_control *control, struct ma86_info *info, const int nrhs, const int idx, ma86pkgtype_d_x[], const ma86pkgtype_d_scale[])`

Definition at line 217 of file HSLLoader.h.

7.125.2.37 `typedef void(* ma86_solve_t)(const int job, const int nrhs, const int idx, ma86pkgtype_d_x, const int order[], void **keep, const struct ma86_control *control, struct ma86_info *info, const ma86pkgtype_d_scale[])`

Definition at line 221 of file HSLLoader.h.

7.125.2.38 `typedef void(* ma86_finalise_t)(void **keep, const struct ma86_control *control)`

Definition at line 225 of file HSLLoader.h.

7.125.2.39 `typedef void(* ma97_default_control_t)(struct ma97_control *control)`

Definition at line 228 of file HSLLoader.h.

7.125.2.40 `typedef void(* ma97_analyse_t)(const int check, const int n, const int ptr[], const int row[], ma97pkgtype_d_val[], void **akeep, const struct ma97_control *control, struct ma97_info *info, int order[])`

Definition at line 229 of file HSLLoader.h.

7.125.2.41 `typedef void(* ma97_factor_t)(const int matrix_type, const int ptr[], const int row[], const ma97pkgtype_d_val[], void **akeep, void **fkeep, const struct ma97_control *control, struct ma97_info *info, const ma97pkgtype_d_scale[])`

Definition at line 232 of file HSLLoader.h.

7.125.2.42 `typedef void(* ma97_factor_solve_t)(const int matrix_type, const int ptr[], const int row[], const ma97pkgtype_d_val[], const int nrhs, ma97pkgtype_d_x[], const int ldx, void **akeep, void **fkeep, const struct ma97_control *control, struct ma97_info *info, const ma97pkgtype_d_scale[])`

Definition at line 236 of file HSLLoader.h.

7.125.2.43 `typedef void(* ma97_solve_t)(const int job, const int nrhs, ma97pkgtype_d_x, const int ldx, void **akeep, void **fkeep, const struct ma97_control *control, struct ma97_info *info)`

Definition at line 241 of file HSLLoader.h.

7.125.2.44 `typedef void(* ma97_finalise_t)(void **akeep, void **fkeep)`

Definition at line 244 of file HSLLoader.h.

7.125.2.45 `typedef void(* ma97_free_akeep_t)(void **akeep)`

Definition at line 245 of file HSLLoader.h.

7.125.2.46 `typedef void(* mc19ad_t)(ipfint *N, ipfint *NZ, double *A, ipfint *IRN, ipfint *ICN, float *R, float *C, float *W)`

Definition at line 247 of file HSLLoader.h.

7.125.2.47 `typedef void(* mc68_default_control_t)(struct mc68_control_i *control)`

Definition at line 249 of file HSLLoader.h.

7.125.2.48 `typedef void(* mc68_order_t)(int ord, int n, const int ptr[], const int row[], int perm[], const struct mc68_control_i *control, struct mc68_info_i *info)`

Definition at line 250 of file HSLLoader.h.

7.125.3 Function Documentation

7.125.3.1 `int LSL_loadHSL (const char * libname, char * msgbuf, int msglen)`

Tries to load a dynamically linked library with HSL routines.

Also tries to load symbols for those HSL routines that are not linked into [lpopt](#), i.e., HAVE_... is not defined. Return a failure if the library cannot be loaded, but not if a symbol is not found.

See Also

[LSL_isMA27available](#)
[LSL_isMA28available](#)
[LSL_isMA57available](#)

[LSL_isMA77available](#)
[LSL_isMA86available](#)
[LSL_isMA97available](#)
[LSL_isMC19available](#)

Parameters

<i>libname</i>	The name under which the HSL lib can be found, or NULL to use a default name (libhsl.SHAR-EDLIBEXT).
<i>msgbuf</i>	A buffer where we can store a failure message. Assumed to be NOT NULL!
<i>msglen</i>	Length of the message buffer.

Returns

Zero on success, nonzero on failure.

7.125.3.2 int LSL_unloadHSL ()

Unloads a loaded HSL library.

Returns

Zero on success, nonzero on failure.

7.125.3.3 int LSL_isHSLLoaded ()

Indicates whether a HSL library has been loaded.

Returns

Zero if not loaded, nonzero if handle is loaded

7.125.3.4 int LSL_isMA27available ()

Indicates whether a HSL library is loaded and all symbols necessary to use MA27 have been found.

Returns

Zero if not available, nonzero if MA27 is available in the loaded library.

7.125.3.5 int LSL_isMA28available ()

Indicates whether a HSL library is loaded and all symbols necessary to use MA28 have been found.

Returns

Zero if not available, nonzero if MA28 is available in the loaded library.

7.125.3.6 int LSL_isMA57available ()

Indicates whether a HSL library is loaded and all symbols necessary to use MA57 have been found.

Returns

Zero if not available, nonzero if MA57 is available in the loaded library.

7.125.3.7 int LSL_isMA77available ()

Indicates whether a HSL library is loaded and all symbols necessary to use MA77 have been found.

Returns

Zero if not available, nonzero if HSL_MA77 is available in the loaded library.

7.125.3.8 int LSL_isMA86available ()

Indicates whether a HSL library is loaded and all symbols necessary to use HSL_MA86 have been found.

Returns

Zero if not available, nonzero if HSL_MA86 is available in the loaded library.

7.125.3.9 int LSL_isMA97available ()

Indicates whether a HSL library is loaded and all symbols necessary to use HSL_MA97 have been found.

Returns

Zero if not available, nonzero if HSL_MA97 is available in the loaded library.

7.125.3.10 int LSL_isMC19available ()

Indicates whether a HSL library is loaded and all symbols necessary to use MA57 have been found.

Returns

Zero if not available, nonzero if MC19 is available in the loaded library.

7.125.3.11 int LSL_isMC68available ()

Indicates whether a HSL library is loaded and all symbols necessary to use HSL_MC68 have been found.

Returns

Zero if not available, nonzero if MC68 is available in the loaded library.

7.125.3.12 char* LSL_HSLLibraryName ()

Returns name of the shared library that should contain HSL.

7.125.3.13 void LSL_setMA27 (ma27ad_t ma27ad, ma27bd_t ma27bd, ma27cd_t ma27cd, ma27id_t ma27id)

sets pointers to MA27 functions

7.125.3.14 void LSL_setMA28 (ma28ad_t ma28ad)

sets pointers to MA28 functions

7.125.3.15 void LSL_setMA57 (ma57ad_t ma57ad, ma57bd_t ma57bd, ma57cd_t ma57cd, ma57ed_t ma57ed, ma57id_t ma57id)

sets pointers to MA57 functions

```
7.125.3.16 void LSL_setMA77 ( ma77_default_control_t ma77_default_control, ma77_open_nelt_t ma77_open_nelt,
ma77_open_t ma77_open, ma77_input_vars_t ma77_input_vars, ma77_input_reals_t ma77_input_reals,
ma77_analyse_t ma77_analyse, ma77_factor_t ma77_factor, ma77_factor_solve_t ma77_factor_solve,
ma77_solve_t ma77_solve, ma77_resid_t ma77_resid, ma77_scale_t ma77_scale, ma77_enquire_posdef_t
ma77_enquire_posdef, ma77_enquire_indef_t ma77_enquire_indef, ma77_alter_t ma77_alter, ma77_restart_t
ma77_restart, ma77_finalise_t ma77_finalise )
```

sets pointers to MA77 functions

```
7.125.3.17 void LSL_setMA86 ( ma86_default_control_t ma86_default_control, ma86_analyse_t ma86_analyse,
ma86_factor_t ma86_factor, ma86_factor_solve_t ma86_factor_solve, ma86_solve_t ma86_solve,
ma86_finalise_t ma86_finalise )
```

sets pointers to MA86 functions

```
7.125.3.18 void LSL_setMA97 ( ma97_default_control_t ma97_default_control, ma97_analyse_t ma97_analyse,
ma97_factor_t ma97_factor, ma97_factor_solve_t ma97_factor_solve, ma97_solve_t ma97_solve,
ma97_finalise_t ma97_finalise, ma97_free_akeep_t ma97_free_akeep )
```

sets pointers to MA97 functions

```
7.125.3.19 void LSL_setMC19 ( mc19ad_t mc19ad )
```

sets pointer to MC19 function

```
7.125.3.20 void LSL_setMC68 ( mc68_default_control_t mc68_default_control, mc68_order_t mc68_order )
```

sets pointers to MC68 functions

7.126 contrib/LinearSolverLoader/LibraryHandler.h File Reference

```
#include "IpoptConfig.h"
#include <unistd.h>
#include <dlfcn.h>
```

Typedefs

- typedef void * [soHandle_t](#)

Functions

- [soHandle_t](#) [LSL_loadLib](#) (const char *libname, char *msgbuf, int msglen)
Loads a dynamically linked library.
- int [LSL_unloadLib](#) ([soHandle_t](#) libhandle)
Unloads a shared library.

7.126.1 Typedef Documentation

7.126.1.1 typedef void* [soHandle_t](#)

Definition at line 27 of file LibraryHandler.h.

7.126.2 Function Documentation

7.126.2.1 `soHandle_t LSL_loadLib (const char * libname, char * msgbuf, int msglen)`

Loads a dynamically linked library.

Parameters

<i>libname</i>	The name of the library to load.
<i>msgbuf</i>	A buffer to store an error message.
<i>msglen</i>	Length of the message buffer.

Returns

Shared library handle, or NULL if failure.

7.126.2.2 `int LSL_unloadLib (soHandle_t libhandle)`

Unloads a shared library.

Parameters

<i>libhandle</i>	Handle of shared library to unload.
------------------	-------------------------------------

Returns

Zero on success, nonzero on failure.

7.127 contrib/LinearSolverLoader/PardisoLoader.h File Reference

Functions

- `int LSL_loadPardisoLib (const char *libname, char *msgbuf, int msglen)`
Tries to load a dynamically linked library with Pardiso.
- `int LSL_unloadPardisoLib ()`
Unloads a loaded Pardiso library.
- `int LSL_isPardisoLoaded ()`
Indicates whether a Pardiso library has been successfully loaded.
- `char * LSL_PardisoLibraryName ()`
Returns name of the shared library that should contain Pardiso.

7.127.1 Function Documentation

7.127.1.1 `int LSL_loadPardisoLib (const char * libname, char * msgbuf, int msglen)`

Tries to load a dynamically linked library with Pardiso.

Return a failure if the library cannot be loaded or not all Pardiso symbols are found.

Parameters

<i>libname</i>	The name under which the Pardiso lib can be found, or NULL to use a default name (libpardiso.-SHAREDLIBEXT).
<i>msgbuf</i>	A buffer where we can store a failure message. Assumed to be NOT NULL!
<i>msglen</i>	Length of the message buffer.

Returns

Zero on success, nonzero on failure.

7.127.1.2 int LSL_unloadPardisoLib ()

Unloads a loaded Pardiso library.

Returns

Zero on success, nonzero on failure.

7.127.1.3 int LSL_isPardisoLoaded ()

Indicates whether a Pardiso library has been successfully loaded.

Returns

Zero if not loaded, nonzero if handle is loaded

7.127.1.4 char* LSL_PardisoLibraryName ()

Returns name of the shared library that should contain Pardiso.

7.128 Interfaces/IpAlgTypes.hpp File Reference

```
#include "IpTypes.hpp"
#include "IpException.hpp"
```

Namespaces

- [Ipopt](#)

Enumerations**Enumerations**

- enum [Ipopt::SolverReturn](#) {
[Ipopt::SUCCESS](#), [Ipopt::MAXITER_EXCEEDED](#), [Ipopt::CPUTIME_EXCEEDED](#), [Ipopt::STOP_AT_TINY_STEP](#),
[Ipopt::STOP_AT_ACCEPTABLE_POINT](#), [Ipopt::LOCAL_INFEASIBILITY](#), [Ipopt::USER_REQUESTED_STOP](#), [Ipopt::FEASIBLE_POINT_FOUND](#),
[Ipopt::DIVERGING_ITERATES](#), [Ipopt::RESTORATION_FAILURE](#), [Ipopt::ERROR_IN_STEP_COMPUTATION](#), [Ipopt::INVALID_NUMBER_DETECTED](#),
[Ipopt::TOO_FEW_DEGREES_OF_FREEDOM](#), [Ipopt::INVALID_OPTION](#), [Ipopt::OUT_OF_MEMORY](#), [Ipopt::INTERNAL_ERROR](#),
[Ipopt::UNASSIGNED](#) }

enum for the return from the optimize algorithm (obviously we need to add more)

Functions

Some exceptions used in multiple places

- [Ipopt::DECLARE_STD_EXCEPTION](#) (LOCALLY_INFEASIBLE)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (TOO_FEW_DOF)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (TINY_STEP_DETECTED)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (ACCEPTABLE_POINT_REACHED)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (FEASIBILITY_PROBLEM_SOLVED)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (INVALID_WARMSTART)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (INTERNAL_ABORT)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (NO_FREE_VARIABLES_BUT_FEASIBLE)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (NO_FREE_VARIABLES_AND_INFEASIBLE)
- [Ipopt::DECLARE_STD_EXCEPTION](#) (FAILED_INITIALIZATION)

Exception FAILED_INITIALIZATION for problem during initialization of a strategy object (or other problems).

7.129 Interfaces/IpInterfacesRegOp.hpp File Reference

```
#include "IpSmartPtr.hpp"
```

Namespaces

- [Ipopt](#)

Functions

- void [Ipopt::RegisterOptions_Interfaces](#) (const SmartPtr< RegisteredOptions > &roptions)

7.130 Interfaces/IpIpoptApplication.hpp File Reference

```
#include <iostream>
#include "IpJournalist.hpp"
#include "IpTNLP.hpp"
#include "IpNLP.hpp"
#include "IpReturnCodes.hpp"
```

Classes

- class [Ipopt::IpoptApplication](#)

This is the main application class for making calls to [Ipopt](#).

Namespaces

- [Ipopt](#)

Macros

- `#define IPOPT_EXPORT(type) type`

Functions

- `Ipopt::DECLARE_STD_EXCEPTION (IPOPT_APPLICATION_ERROR)`
- `IPOPT_EXPORT (class Ipopt::IpoptApplication *) IpoptApplicationFactory()`

7.130.1 Macro Definition Documentation

7.130.1.1 `#define IPOPT_EXPORT(type) type`

Definition at line 20 of file `IpIpoptApplication.hpp`.

7.130.2 Function Documentation

7.130.2.1 `IPOPT_EXPORT (class Ipopt::IpoptApplication *)`

7.131 Interfaces/IpNLP.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpVector.hpp"
#include "IpSmartPtr.hpp"
#include "IpMatrix.hpp"
#include "IpSymMatrix.hpp"
#include "IpOptionsList.hpp"
#include "IpAlgTypes.hpp"
#include "IpReturnCodes.hpp"
```

Classes

- class `Ipopt::NLP`
Brief Class Description.

Namespaces

- `Ipopt`

7.132 Interfaces/IpReturnCodes.h File Reference

```
#include "IpReturnCodes_inc.h"
```

7.133 Interfaces/IpReturnCodes.hpp File Reference

```
#include "IpReturnCodes_inc.h"
```

Namespaces

- [lpopt](#)

Enumerations

- enum [lpopt::ApplicationReturnStatus](#) {
[lpopt::Solve_Succeeded](#) =0, [lpopt::Solved_To_Acceptable_Level](#) =1, [lpopt::Infeasible_Problem_Detected](#) =2,
[lpopt::Search_Direction_Becomes_Too_Small](#) =3,
[lpopt::Diverging_Iterates](#) =4, [lpopt::User_Requested_Stop](#) =5, [lpopt::Feasible_Point_Found](#) =6, [lpopt::Maximum_Iterations_Exceeded](#) =-1,
[lpopt::Restoration_Failed](#) =-2, [lpopt::Error_In_Step_Computation](#) =-3, [lpopt::Maximum_CpuTime_Exceeded](#) =-4,
[lpopt::Not_Enough_Degrees_Of_Freedom](#) =-10,
[lpopt::Invalid_Problem_Definition](#) =-11, [lpopt::Invalid_Option](#) =-12, [lpopt::Invalid_Number_Detected](#) =-13, [lpopt::Unrecoverable_Exception](#) =-100,
[lpopt::Nonlpopt_Exception_Thrown](#) =-101, [lpopt::Insufficient_Memory](#) =-102, [lpopt::Internal_Error](#) =-199 }

Return codes for the Optimize call for an application.

- enum [lpopt::AlgorithmMode](#) { [lpopt::RegularMode](#) =0, [lpopt::RestorationPhaseMode](#) =1 }
enum to indicate the mode in which the algorithm is

7.134 Interfaces/lpReturnCodes_inc.h File Reference

Enumerations

- enum [ApplicationReturnStatus](#) {
[Solve_Succeeded](#) =0, [Solved_To_Acceptable_Level](#) =1, [Infeasible_Problem_Detected](#) =2, [Search_Direction_Becomes_Too_Small](#) =3,
[Diverging_Iterates](#) =4, [User_Requested_Stop](#) =5, [Feasible_Point_Found](#) =6, [Maximum_Iterations_Exceeded](#) =-1,
[Restoration_Failed](#) =-2, [Error_In_Step_Computation](#) =-3, [Maximum_CpuTime_Exceeded](#) =-4, [Not_Enough_Degrees_Of_Freedom](#) =-10,
[Invalid_Problem_Definition](#) =-11, [Invalid_Option](#) =-12, [Invalid_Number_Detected](#) =-13, [Unrecoverable_Exception](#) =-100,
[Nonlpopt_Exception_Thrown](#) =-101, [Insufficient_Memory](#) =-102, [Internal_Error](#) =-199 }

Return codes for the Optimize call for an application.

- enum [AlgorithmMode](#) { [RegularMode](#) =0, [RestorationPhaseMode](#) =1 }
enum to indicate the mode in which the algorithm is

7.134.1 Enumeration Type Documentation

7.134.1.1 enum ApplicationReturnStatus

Return codes for the Optimize call for an application.

Enumerator

Solve_Succeeded

Solved_To_Acceptable_Level

Infeasible_Problem_Detected

Search_Direction_Becomes_Too_Small

Diverging_Iterates
User_Requested_Stop
Feasible_Point_Found
Maximum_Iterations_Exceeded
Restoration_Failed
Error_In_Step_Computation
Maximum_CpuTime_Exceeded
Not_Enough_Degrees_Of_Freedom
Invalid_Problem_Definition
Invalid_Option
Invalid_Number_Detected
Unrecoverable_Exception
Nonlpopt_Exception_Thrown
Insufficient_Memory
Internal_Error

Definition at line 16 of file IpReturnCodes_inc.h.

7.134.1.2 enum **AlgorithmMode**

enum to indicate the mode in which the algorithm is

Enumerator

RegularMode
RestorationPhaseMode

Definition at line 42 of file IpReturnCodes_inc.h.

7.135 Interfaces/lpSolveStatistics.hpp File Reference

```
#include "IpReferenced.hpp"
#include "IpSmartPtr.hpp"
```

Classes

- class [lpopt::SolveStatistics](#)
This class collects statistics about an optimization run, such as iteration count, final infeasibilities etc.

Namespaces

- [lpopt](#)

7.136 Interfaces/lpStdCInterface.h File Reference

```
#include "IpReturnCodes.h"
```

Macros

- #define `IPOPT_EXPORT`(type) type
- #define `TRUE` (1)
- #define `FALSE` (0)

Typedefs

- typedef double `Number`
Type for all number.
- typedef int `Index`
Type for all incides.
- typedef int `Int`
Type for all integers.
- typedef struct lpoptProblemInfo * `lpoptProblem`
Pointer to a `lpopt` Problem.
- typedef int `Bool`
define a boolean type for C
- typedef void * `UserDataPtr`
A pointer for anything that is to be passed between the called and individual callback function.
- typedef `Bool`(* `Eval_F_CB`)(`Index` n, `Number` *x, `Bool` new_x, `Number` *obj_value, `UserDataPtr` user_data)
Type defining the callback function for evaluating the value of the objective function.
- typedef `Bool`(* `Eval_Grad_F_CB`)(`Index` n, `Number` *x, `Bool` new_x, `Number` *grad_f, `UserDataPtr` user_data)
Type defining the callback function for evaluating the gradient of the objective function.
- typedef `Bool`(* `Eval_G_CB`)(`Index` n, `Number` *x, `Bool` new_x, `Index` m, `Number` *g, `UserDataPtr` user_data)
Type defining the callback function for evaluating the value of the constraint functions.
- typedef `Bool`(* `Eval_Jac_G_CB`)(`Index` n, `Number` *x, `Bool` new_x, `Index` m, `Index` nele_jac, `Index` *iRow, `Index` *jCol, `Number` *values, `UserDataPtr` user_data)
Type defining the callback function for evaluating the Jacobian of the constrant functions.
- typedef `Bool`(* `Eval_H_CB`)(`Index` n, `Number` *x, `Bool` new_x, `Number` obj_factor, `Index` m, `Number` *lambda, `Bool` new_lambda, `Index` nele_hess, `Index` *iRow, `Index` *jCol, `Number` *values, `UserDataPtr` user_data)
Type defining the callback function for evaluating the Hessian of the Lagrangian function.
- typedef `Bool`(* `Intermediate_CB`)(`Index` alg_mod, `Index` iter_count, `Number` obj_value, `Number` inf_pr, `Number` inf_du, `Number` mu, `Number` d_norm, `Number` regularization_size, `Number` alpha_du, `Number` alpha_pr, `Index` ls_trials, `UserDataPtr` user_data)
Type defining the callback function for giving intermediate execution control to the user.

Functions

- `IPOPT_EXPORT` (`lpoptProblem`) `CreatelpoptProblem`(`Index` n
Function for creating a new `lpopt` Problem object.
- `IPOPT_EXPORT` (void) `FreelpoptProblem`(`lpoptProblem` lpopt_problem)
Method for freeing a previously created `lpoptProblem`.
- `IPOPT_EXPORT` (`Bool`) `AddlpoptStrOption`(`lpoptProblem` lpopt_problem
Function for adding a string option.
- `IPOPT_EXPORT` (enum `ApplicationReturnStatus`) `lpoptSolve`(`lpoptProblem` lpopt_problem
Function calling the `lpopt` optimization algorithm for a problem previously defined with `CreatelpoptProblem`.

Variables

- [Number * x_L](#)
Lower bounds on variables.
- [Number Number * x_U](#)
Upper bounds on variables.
- [Number Number Index m](#)
Number of constraints.
- [Number Number Index Number * g_L](#)
Lower bounds on constraints.
- [Number Number Index Number Number * g_U](#)
Upper bounds on constraints.
- [Number Number Index Number](#)
[Number Index nele_jac](#)
Number of non-zero elements in constraint Jacobian.
- [Number Number Index Number](#)
[Number Index Index nele_hess](#)
Number of non-zero elements in Hessian of Lagrangian.
- [Number Number Index Number](#)
[Number Index Index Index index_style](#)
indexing style for iRow & jCol, 0 for C style, 1 for Fortran style
- [Number Number Index Number](#)
[Number Index Index Index](#)
[Eval_F_CB eval_f](#)
Callback function for evaluating objective function.
- [Number Number Index Number](#)
[Number Index Index Index](#)
[Eval_F_CB Eval_G_CB eval_g](#)
Callback function for evaluating constraint functions.
- [Number Number Index Number](#)
[Number Index Index Index](#)
[Eval_F_CB Eval_G_CB](#)
[Eval_Grad_F_CB eval_grad_f](#)
Callback function for evaluating gradient of objective function.
- [Number Number Index Number](#)
[Number Index Index Index](#)
[Eval_F_CB Eval_G_CB](#)
[Eval_Grad_F_CB Eval_Jac_G_CB eval_jac_g](#)
Callback function for evaluating Jacobian of constraint functions.
- [Number Number Index Number](#)
[Number Index Index Index](#)
[Eval_F_CB Eval_G_CB](#)
[Eval_Grad_F_CB Eval_Jac_G_CB](#)
[Eval_H_CB eval_h](#)
Callback function for evaluating Hessian of Lagrangian function.
- [char * keyword](#)
- [char char * val](#)
- [char * file_name](#)
- [char Int print_level](#)
- [Number obj_scaling](#)

- [Number](#) [Number](#) * [x_scaling](#)
- [Number](#) [Number](#) [Number](#) * [g_scaling](#)
- [Intermediate_CB](#) [intermediate_cb](#)
- [Number](#) * [x](#)
Input: Starting point Output: Optimal solution.
- [Number](#) [Number](#) * [g](#)
Values of constraint at final point (output only - ignored if set to NULL)
- [Number](#) [Number](#) [Number](#) * [obj_val](#)
Final value of objective function (output only - ignored if set to NULL)
- [Number](#) [Number](#) [Number](#) [Number](#) * [mult_g](#)
Input: Initial values for the constraint multipliers (only if warm start option is chosen) Output: Final multipliers for constraints (ignored if set to NULL)
- [Number](#) [Number](#) [Number](#) [Number](#) [Number](#) * [mult_x_L](#)
Input: Initial values for the multipliers for lower variable bounds (only if warm start option is chosen) Output: Final multipliers for lower variable bounds (ignored if set to NULL)
- [Number](#) [Number](#) [Number](#) [Number](#) [Number](#) * [mult_x_U](#)
Input: Initial values for the multipliers for upper variable bounds (only if warm start option is chosen) Output: Final multipliers for upper variable bounds (ignored if set to NULL)
- [Number](#) [Number](#) [Number](#) [Number](#) [Number](#) [Number](#) [UserDataPtr](#) [user_data](#)
Pointer to user data.

7.136.1 Macro Definition Documentation

7.136.1.1 `#define IPOPT_EXPORT(type) type`

Definition at line 22 of file `lpStdCInterface.h`.

7.136.1.2 `#define TRUE (1)`

Definition at line 57 of file `lpStdCInterface.h`.

7.136.1.3 `#define FALSE (0)`

Definition at line 60 of file `lpStdCInterface.h`.

7.136.2 Typedef Documentation

7.136.2.1 `typedef double Number`

Type for all number.

We need to make sure that this is identical with what is defined in [Common/lpTypes.hpp](#)

Definition at line 33 of file `lpStdCInterface.h`.

7.136.2.2 `typedef int Index`

Type for all indices.

We need to make sure that this is identical with what is defined in [Common/lpTypes.hpp](#)

Definition at line 37 of file `lpStdCInterface.h`.

7.136.2.3 typedef int Int

Type for all integers.

We need to make sure that this is identical with what is defined in [Common/lpTypes.hpp](#)

Definition at line 41 of file lpStdCInterface.h.

7.136.2.4 typedef struct lpoptProblemInfo* lpoptProblem

Pointer to a [lpopt](#) Problem.

Definition at line 52 of file lpStdCInterface.h.

7.136.2.5 typedef int Bool

define a boolean type for C

Definition at line 55 of file lpStdCInterface.h.

7.136.2.6 typedef void* UserDataPtr

A pointer for anything that is to be passed between the called and individual callback function.

Definition at line 65 of file lpStdCInterface.h.

7.136.2.7 typedef Bool(* Eval_F_CB)(Index n, Number *x, Bool new_x, Number *obj_value, UserDataPtr user_data)

Type defining the callback function for evaluating the value of the objective function.

Return value should be set to false if there was a problem doing the evaluation.

Definition at line 70 of file lpStdCInterface.h.

7.136.2.8 typedef Bool(* Eval_Grad_F_CB)(Index n, Number *x, Bool new_x, Number *grad_f, UserDataPtr user_data)

Type defining the callback function for evaluating the gradient of the objective function.

Return value should be set to false if there was a problem doing the evaluation.

Definition at line 76 of file lpStdCInterface.h.

7.136.2.9 typedef Bool(* Eval_G_CB)(Index n, Number *x, Bool new_x, Index m, Number *g, UserDataPtr user_data)

Type defining the callback function for evaluating the value of the constraint functions.

Return value should be set to false if there was a problem doing the evaluation.

Definition at line 82 of file lpStdCInterface.h.

7.136.2.10 typedef Bool(* Eval_Jac_G_CB)(Index n, Number *x, Bool new_x, Index m, Index nele_jac, Index *iRow, Index *jCol, Number *values, UserDataPtr user_data)

Type defining the callback function for evaluating the Jacobian of the constraint functions.

Return value should be set to false if there was a problem doing the evaluation.

Definition at line 88 of file lpStdCInterface.h.

7.136.2.11 typedef Bool(* Eval_H_CB)(Index n, Number *x, Bool new_x, Number obj_factor, Index m, Number *lambda, Bool new_lambda, Index nele_hess, Index *iRow, Index *jCol, Number *values, UserDataPtr user_data)

Type defining the callback function for evaluating the Hessian of the Lagrangian function.

Return value should be set to false if there was a problem doing the evaluation.

Definition at line 96 of file lpStdCInterface.h.

7.136.2.12 `typedef Bool>(* Intermediate_CB)(Index alg_mod, Index iter_count, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, UserDataPtr user_data)`

Type defining the callback function for giving intermediate execution control to the user.

If set, it is called once per iteration, providing the user with some information on the state of the optimization. This can be used to print some user-defined output. It also gives the user a way to terminate the optimization prematurely. If this method returns false, `lpopt` will terminate the optimization.

Definition at line 108 of file lpStdCInterface.h.

7.136.3 Function Documentation

7.136.3.1 IPOPT_EXPORT (lpoptProblem)

Function for creating a new `lpopt` Problem object.

This function returns an object that can be passed to the `lpoptSolve` call. It contains the basic definition of the optimization problem, such as number of variables and constraints, bounds on variables and constraints, information about the derivatives, and the callback function for the computation of the optimization problem functions and derivatives. During this call, the options file `PARAMS.DAT` is read as well.

If `NULL` is returned, there was a problem with one of the inputs or reading the options file. Number of optimization variables

7.136.3.2 IPOPT_EXPORT (void)

Method for freeing a previously created `lpoptProblem`.

After freeing an `lpoptProblem`, it cannot be used anymore.

7.136.3.3 IPOPT_EXPORT (Bool)

Function for adding a string option.

Setting a callback function for the "intermediate callback" method in the `TNLP`.

Optional function for setting scaling parameter for the `NLP`.

Function for opening an output file for a given name with given printlevel.

Function for adding an `Int` option.

Function for adding a `Number` option.

Returns `FALSE` the option could not be set (e.g., if keyword is unknown)

Returns false, if there was a problem opening the file.

This corresponds to the `get_scaling_parameters` method in `TNLP`. If the pointers `x_scaling` or `g_scaling` are `NULL`, then no scaling for `x` resp. `g` is done.

This gives control back to the user once per iteration. If set, it provides the user with some information on the state of the optimization. This can be used to print some user-defined output. It also gives the user a way to terminate the optimization prematurely. If the callback method returns false, `lpopt` will terminate the optimization. Calling this set method to set the `CB` pointer to `NULL` disables the intermediate callback functionality.

7.136.3.4 IPOPT_EXPORT (enum *ApplicationReturnStatus*)

Function calling the [lpopt](#) optimization algorithm for a problem previously defined with CreateIpoptProblem.

The return specified outcome of the optimization procedure (e.g., success, failure etc). Problem that is to be optimized. [lpopt](#) will use the options previously specified with AddIpoptOption (etc) for this problem.

7.136.4 Variable Documentation

7.136.4.1 Number* x_L

Lower bounds on variables.

This array of size n is copied internally, so that the caller can change the incoming data after return without that Ipopt-Problem is modified. Any value less or equal than the number specified by option 'nlp_lower_bound_inf' is interpreted to be minus infinity.

Definition at line 137 of file IpStdCInterface.h.

7.136.4.2 Number Number* x_U

Upper bounds on variables.

This array of size n is copied internally, so that the caller can change the incoming data after return without that Ipopt-Problem is modified. Any value greater or equal than the number specified by option 'nlp_upper_bound_inf' is interpreted to be plus infinity.

Definition at line 137 of file IpStdCInterface.h.

7.136.4.3 Number Number Index m

Number of constraints.

Definition at line 137 of file IpStdCInterface.h.

7.136.4.4 Number Number Index Number* g_L

Lower bounds on constraints.

This array of size m is copied internally, so that the caller can change the incoming data after return without that Ipopt-Problem is modified. Any value less or equal than the number specified by option 'nlp_lower_bound_inf' is interpreted to be minus infinity.

Definition at line 137 of file IpStdCInterface.h.

7.136.4.5 Number Number Index Number Number* g_U

Upper bounds on constraints.

This array of size m is copied internally, so that the caller can change the incoming data after return without that Ipopt-Problem is modified. Any value greater or equal than the number specified by option 'nlp_upper_bound_inf' is interpreted to be plus infinity.

Definition at line 137 of file IpStdCInterface.h.

7.136.4.6 Number Number Index Number Number Index nele_jac

Number of non-zero elements in constraint Jacobian.

Definition at line 137 of file IpStdCInterface.h.

7.136.4.7 Number Number Index Number Number Index Index nele_hess

Number of non-zero elements in Hessian of Lagrangian.

Definition at line 137 of file lpStdCInterface.h.

7.136.4.8 Number Number Index Number Number Index Index Index index_style

indexing style for iRow & jCol, 0 for C style, 1 for Fortran style

Definition at line 137 of file lpStdCInterface.h.

7.136.4.9 Number Number Index Number Number Index Index Index Eval_F_CB eval_f

Callback function for evaluating objective function.

Definition at line 137 of file lpStdCInterface.h.

7.136.4.10 Number Number Index Number Number Index Index Index Eval_F_CB Eval_G_CB eval_g

Callback function for evaluating constraint functions.

Definition at line 137 of file lpStdCInterface.h.

7.136.4.11 Number Number Index Number Number Index Index Index Eval_F_CB Eval_G_CB Eval_Grad_F_CB eval_grad_f

Callback function for evaluating gradient of objective function.

Definition at line 137 of file lpStdCInterface.h.

7.136.4.12 Number Number Index Number Number Index Index Index Eval_F_CB Eval_G_CB Eval_Grad_F_CB Eval_Jac_G_CB eval_jac_g

Callback function for evaluating Jacobian of constraint functions.

Definition at line 137 of file lpStdCInterface.h.

7.136.4.13 Number Number Index Number Number Index Index Index Eval_F_CB Eval_G_CB Eval_Grad_F_CB Eval_Jac_G_CB Eval_H_CB eval_h

Callback function for evaluating Hessian of Lagrangian function.

Definition at line 137 of file lpStdCInterface.h.

7.136.4.14 char * keyword

Definition at line 189 of file lpStdCInterface.h.

7.136.4.15 char Int val

Definition at line 189 of file lpStdCInterface.h.

7.136.4.16 char* file_name

Definition at line 202 of file lpStdCInterface.h.

7.136.4.17 char Int print_level

Definition at line 202 of file lpStdCInterface.h.

7.136.4.18 Number obj_scaling

Definition at line 210 of file IpStdCInterface.h.

7.136.4.19 Number Number* x_scaling

Definition at line 210 of file IpStdCInterface.h.

7.136.4.20 Number Number Number* g_scaling

Definition at line 210 of file IpStdCInterface.h.

7.136.4.21 Intermediate_CB intermediate_cb

Definition at line 224 of file IpStdCInterface.h.

7.136.4.22 Number* x

Input: Starting point Output: Optimal solution.

Definition at line 238 of file IpStdCInterface.h.

7.136.4.23 Number Number* g

Values of constraint at final point (output only - ignored if set to NULL)

Definition at line 238 of file IpStdCInterface.h.

7.136.4.24 Number Number Number* obj_val

Final value of objective function (output only - ignored if set to NULL)

Definition at line 238 of file IpStdCInterface.h.

7.136.4.25 Number Number Number Number* mult_g

Input: Initial values for the constraint multipliers (only if warm start option is chosen) Output: Final multipliers for constraints (ignored if set to NULL)

Definition at line 238 of file IpStdCInterface.h.

7.136.4.26 Number Number Number Number Number* mult_x_L

Input: Initial values for the multipliers for lower variable bounds (only if warm start option is chosen) Output: Final multipliers for lower variable bounds (ignored if set to NULL)

Definition at line 238 of file IpStdCInterface.h.

7.136.4.27 Number Number Number Number Number Number* mult_x_U

Input: Initial values for the multipliers for upper variable bounds (only if warm start option is chosen) Output: Final multipliers for upper variable bounds (ignored if set to NULL)

Definition at line 238 of file IpStdCInterface.h.

7.136.4.28 Number Number Number Number Number Number UserDataPtr user_data

Pointer to user data.

This will be passed unmodified to the callback functions.

Definition at line 238 of file IpStdCInterface.h.

7.137 Interfaces/IpStdInterfaceTNLP.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpTNLP.hpp"
#include "IpJournalist.hpp"
#include "IpException.hpp"
#include "IpStdCInterface.h"
#include "IpSmartPtr.hpp"
```

Classes

- class [Ipopt::StdInterfaceTNLP](#)
Implementation of a [TNLP](#) for the Standard C interface.

Namespaces

- [Ipopt](#)

Functions

- [Ipopt::DECLARE_STD_EXCEPTION](#) (INVALID_STDINTERFACE_NLP)
Declare exception that is thrown when invalid [NLP](#) data is provided.

7.138 Interfaces/IpTNLP.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpReferenced.hpp"
#include "IpException.hpp"
#include "IpAlgTypes.hpp"
#include "IpReturnCodes.hpp"
#include <map>
```

Classes

- class [Ipopt::TNLP](#)
Base class for all [NLP](#)'s that use standard triplet matrix form and dense vectors.

Namespaces

- [Ipopt](#)

7.139 Interfaces/IpTNLPAdapter.hpp File Reference

```
#include "IpNLP.hpp"
#include "IpTNLP.hpp"
#include "IpOrigIpoptNLP.hpp"
#include <list>
```

Classes

- class [Ipopt::TNLPAdapter](#)
This class Adapts the [TNLP](#) interface so it looks like an [NLP](#) interface.

Namespaces

- [Ipopt](#)

7.140 Interfaces/IpTNLPReducer.hpp File Reference

```
#include "IpTNLP.hpp"
```

Classes

- class [Ipopt::TNLPReducer](#)
This is a wrapper around a given [TNLP](#) class that takes out a list of constraints that are given to the constructor.

Namespaces

- [Ipopt](#)

7.141 LinAlg/IpBlas.hpp File Reference

```
#include "IpUtils.hpp"
```

Namespaces

- [Ipopt](#)

Functions

- [Number Ipopt::IpBlasDdot](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX, const [Number](#) *y, [Index](#) incY)
Wrapper for BLAS function DDOT.
- [Number Ipopt::IpBlasDnrm2](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX)
Wrapper for BLAS function DNRM2.
- [Number Ipopt::IpBlasDasum](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX)

Wrapper for BLAS function DASUM.

- [Index Ipopt::IpBlasIdamax](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX)

Wrapper for BLAS function IDAMAX.

- void [Ipopt::IpBlasDcopy](#) ([Index](#) size, const [Number](#) *x, [Index](#) incX, [Number](#) *y, [Index](#) incY)

Wrapper for BLAS subroutine DCOPY.

- void [Ipopt::IpBlasDaxpy](#) ([Index](#) size, [Number](#) alpha, const [Number](#) *x, [Index](#) incX, [Number](#) *y, [Index](#) incY)

Wrapper for BLAS subroutine DAXPY.

- void [Ipopt::IpBlasDscal](#) ([Index](#) size, [Number](#) alpha, [Number](#) *x, [Index](#) incX)

Wrapper for BLAS subroutine DSCAL.

- void [Ipopt::IpBlasDgemv](#) (bool trans, [Index](#) nRows, [Index](#) nCols, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, const [Number](#) *x, [Index](#) incX, [Number](#) beta, [Number](#) *y, [Index](#) incY)

Wrapper for BLAS subroutine DGEMV.

- void [Ipopt::IpBlasDsymv](#) ([Index](#) n, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, const [Number](#) *x, [Index](#) incX, [Number](#) beta, [Number](#) *y, [Index](#) incY)

Wrapper for BLAS subroutine DSYMV.

- void [Ipopt::IpBlasDgemm](#) (bool transa, bool transb, [Index](#) m, [Index](#) n, [Index](#) k, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, const [Number](#) *B, [Index](#) ldB, [Number](#) beta, [Number](#) *C, [Index](#) ldC)

Wrapper for BLAS subroutine DGEMM.

- void [Ipopt::IpBlasDsyrk](#) (bool trans, [Index](#) ndim, [Index](#) nrank, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, [Number](#) beta, [Number](#) *C, [Index](#) ldC)

Wrapper for BLAS subroutine DSYRK.

- void [Ipopt::IpBlasDtrsm](#) (bool trans, [Index](#) ndim, [Index](#) nrhs, [Number](#) alpha, const [Number](#) *A, [Index](#) ldA, [Number](#) *B, [Index](#) ldB)

Wrapper for BLAS subroutine DTRSM.

7.142 LinAlg/IpCompoundMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
```

Classes

- class [Ipopt::CompoundMatrix](#)
Class for Matrices consisting of other matrices.
- class [Ipopt::CompoundMatrixSpace](#)
This is the matrix space for [CompoundMatrix](#).

Namespaces

- [Ipopt](#)

7.143 LinAlg/IpCompoundSymMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
```

Classes

- class [Ipopt::CompoundSymMatrix](#)
Class for symmetric matrices consisting of other matrices.
- class [Ipopt::CompoundSymMatrixSpace](#)
This is the matrix space for [CompoundSymMatrix](#).

Namespaces

- [Ipopt](#)

7.144 LinAlg/IpCompoundVector.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpVector.hpp"
#include <vector>
```

Classes

- class [Ipopt::CompoundVector](#)
Class of Vectors consisting of other vectors.
- class [Ipopt::CompoundVectorSpace](#)
This vectors space is the vector space for [CompoundVector](#).

Namespaces

- [Ipopt](#)

7.145 LinAlg/IpDenseGenMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
#include "IpDenseVector.hpp"
#include "IpDenseSymMatrix.hpp"
```

Classes

- class [Ipopt::DenseGenMatrix](#)
Class for dense general matrices.
- class [Ipopt::DenseGenMatrixSpace](#)
This is the matrix space for [DenseGenMatrix](#).

Namespaces

- [Ipopt](#)

7.146 LinAlg/IpDenseSymMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
#include "IpMultiVectorMatrix.hpp"
#include "IpDenseVector.hpp"
```

Classes

- class [Ipopt::DenseSymMatrix](#)
Class for dense symetrix matrices.
- class [Ipopt::DenseSymMatrixSpace](#)
This is the matrix space for [DenseSymMatrix](#).

Namespaces

- [Ipopt](#)

7.147 LinAlg/IpDenseVector.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpVector.hpp"
#include <map>
```

Classes

- class [Ipopt::DenseVector](#)
Dense [Vector](#) Implementation.
- class [Ipopt::DenseVectorSpace](#)
This vectors space is the vector space for [DenseVector](#).

Namespaces

- [Ipopt](#)

Typedefs

- typedef std::map< std::string, std::vector< std::string > > [Ipopt::StringMetaDataMapType](#)
typedefs for the map variables that define meta data for the [DenseVectorSpace](#)
- typedef std::map< std::string, std::vector< [Index](#) > > [Ipopt::IntegerMetaDataMapType](#)
- typedef std::map< std::string, std::vector< [Number](#) > > [Ipopt::NumericMetaDataMapType](#)

Functions

Exceptions

- [Ipopt::DECLARE_STD_EXCEPTION](#) (METADATA_ERROR)

7.148 LinAlg/IpDiagMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
```

Classes

- class [Ipopt::DiagMatrix](#)
Class for diagonal matrices.
- class [Ipopt::DiagMatrixSpace](#)
This is the matrix space for [DiagMatrix](#).

Namespaces

- [Ipopt](#)

7.149 LinAlg/IpExpandedMultiVectorMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
#include "IpExpansionMatrix.hpp"
```

Classes

- class [Ipopt::ExpandedMultiVectorMatrix](#)
Class for Matrices with few rows that consists of Vectors, together with a premultiplied Expansion matrix.
- class [Ipopt::ExpandedMultiVectorMatrixSpace](#)
This is the matrix space for [ExpandedMultiVectorMatrix](#).

Namespaces

- [Ipopt](#)

7.150 LinAlg/IpExpansionMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
```

Classes

- class [lpopt::ExpansionMatrix](#)
Class for expansion/projection matrices.
- class [lpopt::ExpansionMatrixSpace](#)
This is the matrix space for [ExpansionMatrix](#).

Namespaces

- [lpopt](#)

7.151 LinAlg/lpIdentityMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
```

Classes

- class [lpopt::IdentityMatrix](#)
Class for Matrices which are multiples of the identity matrix.
- class [lpopt::IdentityMatrixSpace](#)
This is the matrix space for [IdentityMatrix](#).

Namespaces

- [lpopt](#)

7.152 LinAlg/lpLapack.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpException.hpp"
```

Namespaces

- [lpopt](#)

Functions

- [lpopt::DECLARE_STD_EXCEPTION](#) (LAPACK_NOT_INCLUDED)
- void [lpopt::lpLapackDpotrs](#) (Index ndim, Index nrhs, const Number *a, Index lda, Number *b, Index ldb)
Wrapper for LAPACK subroutine DPOTRS.
- void [lpopt::lpLapackDpotrf](#) (Index ndim, Number *a, Index lda, Index &info)
Wrapper for LAPACK subroutine DPOTRF.
- void [lpopt::lpLapackDsyev](#) (bool compute_eigenvalues, Index ndim, Number *a, Index lda, Number *w, Index &info)

Wrapper for LAPACK subroutine DSYEV.

- void [Ipopt::IpLapackDgetrf](#) ([Index](#) ndim, [Number](#) *a, [Index](#) *pivot, [Index](#) lda, [Index](#) &info)

Wrapper for LAPACK subroutine DGETRF.

- void [Ipopt::IpLapackDgetrs](#) ([Index](#) ndim, [Index](#) nrhs, const [Number](#) *a, [Index](#) lda, [Index](#) *ipiv, [Number](#) *b, [Index](#) ldb)

Wrapper for LAPACK subroutine DGETRS.

7.153 LinAlg/IpLowRankUpdateSymMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
#include "IpMultiVectorMatrix.hpp"
```

Classes

- class [Ipopt::LowRankUpdateSymMatrix](#)
Class for symmetric matrices, represented as low-rank updates.
- class [Ipopt::LowRankUpdateSymMatrixSpace](#)
This is the matrix space for [LowRankUpdateSymMatrix](#).

Namespaces

- [Ipopt](#)

7.154 LinAlg/IpMatrix.hpp File Reference

```
#include "IpVector.hpp"
```

Classes

- class [Ipopt::Matrix](#)
[Matrix](#) Base Class.
- class [Ipopt::MatrixSpace](#)
[MatrixSpace](#) base class, corresponding to the [Matrix](#) base class.

Namespaces

- [Ipopt](#)

Macros

- `#define DBG_PRINT_MATRIX(__verbose_level, __mat_name, __mat)`

7.154.1 Macro Definition Documentation

7.154.1.1 `#define DBG_PRINT_MATRIX(__verbose_level, __mat_name, __mat)`

Definition at line 330 of file IpMatrix.hpp.

7.155 LinAlg/IpMultiVectorMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
```

Classes

- class [Ipopt::MultiVectorMatrix](#)
Class for Matrices with few columns that consists of Vectors.
- class [Ipopt::MultiVectorMatrixSpace](#)
This is the matrix space for [MultiVectorMatrix](#).

Namespaces

- [Ipopt](#)

7.156 LinAlg/IpScaledMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
```

Classes

- class [Ipopt::ScaledMatrix](#)
Class for a [Matrix](#) in conjunction with its scaling factors for row and column scaling.
- class [Ipopt::ScaledMatrixSpace](#)
This is the matrix space for [ScaledMatrix](#).

Namespaces

- [Ipopt](#)

7.157 LinAlg/IpSumMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
```

Classes

- class [Ipopt::SumMatrix](#)
Class for Matrices which are sum of matrices.
- class [Ipopt::SumMatrixSpace](#)
Class for matrix space for [SumMatrix](#).

Namespaces

- [Ipopt](#)

7.158 LinAlg/IpSumSymMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
```

Classes

- class [Ipopt::SumSymMatrix](#)
Class for Matrices which are sum of symmetric matrices.
- class [Ipopt::SumSymMatrixSpace](#)
Class for matrix space for [SumSymMatrix](#).

Namespaces

- [Ipopt](#)

7.159 LinAlg/IpSymMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
```

Classes

- class [Ipopt::SymMatrix](#)
This is the base class for all derived symmetric matrix types.
- class [Ipopt::SymMatrixSpace](#)
[SymMatrixSpace](#) base class, corresponding to the [SymMatrix](#) base class.

Namespaces

- [Ipopt](#)

7.160 LinAlg/IpSymScaledMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
```

Classes

- class [Ipopt::SymScaledMatrix](#)
Class for a [Matrix](#) in conjunction with its scaling factors for row and column scaling.
- class [Ipopt::SymScaledMatrixSpace](#)
This is the matrix space for [SymScaledMatrix](#).

Namespaces

- [Ipopt](#)

7.161 LinAlg/IpTransposeMatrix.hpp File Reference

```
#include "IpMatrix.hpp"
```

Classes

- class [Ipopt::TransposeMatrix](#)
Class for Matrices which are the transpose of another matrix.
- class [Ipopt::TransposeMatrixSpace](#)
This is the matrix space for [TransposeMatrix](#).

Namespaces

- [Ipopt](#)

7.162 LinAlg/IpVector.hpp File Reference

```
#include "IpTypes.hpp"
#include "IpTaggedObject.hpp"
#include "IpCachedResults.hpp"
#include "IpSmartPtr.hpp"
#include "IpJournalist.hpp"
#include "IpException.hpp"
#include <vector>
```

Classes

- class [Ipopt::Vector](#)
Vector Base Class.
- class [Ipopt::VectorSpace](#)
VectorSpace base class, corresponding to the [Vector](#) base class.

Namespaces

- [Ipopt](#)

Macros

- `#define DBG_PRINT_VECTOR(__verbose_level, __vec_name, __vec)`

Functions

- [Ipopt::DECLARE_STD_EXCEPTION](#) (UNIMPLEMENTED_LINALG_METHOD_CALLED)
Exception that can be used to flag unimplemented linear algebra methods.

7.162.1 Macro Definition Documentation

7.162.1.1 `#define DBG_PRINT_VECTOR(__verbose_level, __vec_name, __vec)`

Definition at line 753 of file `IpVector.hpp`.

7.163 LinAlg/IpZeroMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpMatrix.hpp"
```

Classes

- class [Ipopt::ZeroMatrix](#)
Class for Matrices with only zero entries.
- class [Ipopt::ZeroMatrixSpace](#)
Class for matrix space for [ZeroMatrix](#).

Namespaces

- [Ipopt](#)

7.164 LinAlg/IpZeroSymMatrix.hpp File Reference

```
#include "IpUtils.hpp"
#include "IpSymMatrix.hpp"
```

Classes

- class [lpopt::ZeroSymMatrix](#)
Class for Symmetric Matrices with only zero entries.
- class [lpopt::ZeroSymMatrixSpace](#)
Class for matrix space for [ZeroSymMatrix](#).

Namespaces

- [lpopt](#)

7.165 LinAlg/TMatrices/lpGenTMatrix.hpp File Reference

```
#include "IpUtils.hpp"  
#include "IpMatrix.hpp"
```

Classes

- class [lpopt::GenTMatrix](#)
Class for general matrices stored in triplet format.
- class [lpopt::GenTMatrixSpace](#)
This is the matrix space for a [GenTMatrix](#) with fixed sparsity structure.

Namespaces

- [lpopt](#)

7.166 LinAlg/TMatrices/lpSymTMatrix.hpp File Reference

```
#include "IpUtils.hpp"  
#include "IpSymMatrix.hpp"
```

Classes

- class [lpopt::SymTMatrix](#)
Class for symmetric matrices stored in triplet format.
- class [lpopt::SymTMatrixSpace](#)
This is the matrix space for a [SymTMatrix](#) with fixed sparsity structure.

Namespaces

- [lpopt](#)

7.167 LinAlg/TMatrices/lpTripletHelper.hpp File Reference

```
#include "IpTypes.hpp"  
#include "IpException.hpp"
```

Classes

- class [lpopt::TripletHelper](#)

Namespaces

- [lpopt](#)

Functions

- [lpopt::DECLARE_STD_EXCEPTION](#) (UNKNOWN_MATRIX_TYPE)
- [lpopt::DECLARE_STD_EXCEPTION](#) (UNKNOWN_VECTOR_TYPE)

Index

Symbols

- ~AdaptiveMuUpdate
 - Ipopt::AdaptiveMuUpdate, [59](#)
- ~AlgorithmBuilder
 - Ipopt::AlgorithmBuilder, [64](#)
- ~AlgorithmStrategyObject
 - Ipopt::AlgorithmStrategyObject, [67](#)
- ~AmplOption
 - Ipopt::AmplOptionsList::AmplOption, [71](#)
- ~AmplOptionsList
 - Ipopt::AmplOptionsList, [73](#)
- ~AmplSuffixHandler
 - Ipopt::AmplSuffixHandler, [76](#)
- ~AmplTNLP
 - Ipopt::AmplTNLP, [81](#)
- ~AugRestoSystemSolver
 - Ipopt::AugRestoSystemSolver, [90](#)
- ~AugSystemSolver
 - Ipopt::AugSystemSolver, [94](#)
- ~BacktrackingLSAceptor
 - Ipopt::BacktrackingLSAceptor, [108](#)
- ~BacktrackingLineSearch
 - Ipopt::BacktrackingLineSearch, [100](#)
- ~CGPenaltyCq
 - Ipopt::CGPenaltyCq, [119](#)
- ~CGPenaltyData
 - Ipopt::CGPenaltyData, [124](#)
- ~CGPenaltyLSAceptor
 - Ipopt::CGPenaltyLSAceptor, [132](#)
- ~CGPerturbationHandler
 - Ipopt::CGPerturbationHandler, [142](#)
- ~CGSearchDirCalculator
 - Ipopt::CGSearchDirCalculator, [148](#)
- ~CachedResults
 - Ipopt::CachedResults, [114](#)
- ~CompoundMatrix
 - Ipopt::CompoundMatrix, [153](#)
- ~CompoundMatrixSpace
 - Ipopt::CompoundMatrixSpace, [158](#)
- ~CompoundSymMatrix
 - Ipopt::CompoundSymMatrix, [162](#)
- ~CompoundSymMatrixSpace
 - Ipopt::CompoundSymMatrixSpace, [167](#)
- ~CompoundVector
 - Ipopt::CompoundVector, [171](#)
- ~CompoundVectorSpace
 - Ipopt::CompoundVectorSpace, [177](#)
- ~ConvergenceCheck
 - Ipopt::ConvergenceCheck, [181](#)
- ~DefaultIterateInitializer
 - Ipopt::DefaultIterateInitializer, [185](#)
- ~DenseGenMatrix
 - Ipopt::DenseGenMatrix, [190](#)
- ~DenseGenMatrixSpace
 - Ipopt::DenseGenMatrixSpace, [195](#)
- ~DenseSymMatrix
 - Ipopt::DenseSymMatrix, [197](#)
- ~DenseSymMatrixSpace
 - Ipopt::DenseSymMatrixSpace, [200](#)
- ~DenseVector
 - Ipopt::DenseVector, [204](#)
- ~DenseVectorSpace
 - Ipopt::DenseVectorSpace, [211](#)
- ~DependentResult
 - Ipopt::DependentResult, [214](#)
- ~DiagMatrix
 - Ipopt::DiagMatrix, [218](#)
- ~DiagMatrixSpace
 - Ipopt::DiagMatrixSpace, [220](#)
- ~EqMultiplierCalculator
 - Ipopt::EqMultiplierCalculator, [222](#)
- ~EquilibrationScaling
 - Ipopt::EquilibrationScaling, [224](#)
- ~ExactHessianUpdater
 - Ipopt::ExactHessianUpdater, [227](#)
- ~ExpandedMultiVectorMatrix
 - Ipopt::ExpandedMultiVectorMatrix, [229](#)
- ~ExpandedMultiVectorMatrixSpace
 - Ipopt::ExpandedMultiVectorMatrixSpace, [232](#)
- ~ExpansionMatrix
 - Ipopt::ExpansionMatrix, [235](#)
- ~ExpansionMatrixSpace
 - Ipopt::ExpansionMatrixSpace, [238](#)
- ~FileJournal
 - Ipopt::FileJournal, [241](#)
- ~Filter
 - Ipopt::Filter, [243](#)
- ~FilterEntry
 - Ipopt::FilterEntry, [245](#)
- ~FilterLSAceptor
 - Ipopt::FilterLSAceptor, [250](#)
- ~GenAugSystemSolver
 - Ipopt::GenAugSystemSolver, [258](#)
- ~GenKKTsSolverInterface
 - Ipopt::GenKKTsSolverInterface, [262](#)
- ~GenTMatrix
 - Ipopt::GenTMatrix, [266](#)
- ~GenTMatrixSpace
 - Ipopt::GenTMatrixSpace, [270](#)
- ~GradientScaling
 - Ipopt::GradientScaling, [273](#)
- ~HessianUpdater

- Ipopt::HessianUpdater, [276](#)
- ~IdentityMatrix
 - Ipopt::IdentityMatrix, [278](#)
- ~IdentityMatrixSpace
 - Ipopt::IdentityMatrixSpace, [281](#)
- ~InexactAlgorithmBuilder
 - Ipopt::InexactAlgorithmBuilder, [283](#)
- ~InexactCq
 - Ipopt::InexactCq, [286](#)
- ~InexactData
 - Ipopt::InexactData, [291](#)
- ~InexactDoglegNormalStep
 - Ipopt::InexactDoglegNormalStep, [295](#)
- ~InexactLSAcceptor
 - Ipopt::InexactLSAcceptor, [300](#)
- ~InexactNewtonNormalStep
 - Ipopt::InexactNewtonNormalStep, [306](#)
- ~InexactNormalStepCalculator
 - Ipopt::InexactNormalStepCalculator, [309](#)
- ~InexactNormalTerminationTester
 - Ipopt::InexactNormalTerminationTester, [311](#)
- ~InexactPDSolver
 - Ipopt::InexactPDSolver, [315](#)
- ~InexactPDTerminationTester
 - Ipopt::InexactPDTerminationTester, [319](#)
- ~InexactSearchDirCalculator
 - Ipopt::InexactSearchDirCalculator, [325](#)
- ~InexactTSymScalingMethod
 - Ipopt::InexactTSymScalingMethod, [328](#)
- ~IpoptAdditionalCq
 - Ipopt::IpoptAdditionalCq, [330](#)
- ~IpoptAdditionalData
 - Ipopt::IpoptAdditionalData, [331](#)
- ~IpoptAlgorithm
 - Ipopt::IpoptAlgorithm, [335](#)
- ~IpoptApplication
 - Ipopt::IpoptApplication, [341](#)
- ~IpoptCalculatedQuantities
 - Ipopt::IpoptCalculatedQuantities, [355](#)
- ~IpoptData
 - Ipopt::IpoptData, [379](#)
- ~IpoptException
 - Ipopt::IpoptException, [390](#)
- ~IpoptNLP
 - Ipopt::IpoptNLP, [394](#)
- ~IterateInitializer
 - Ipopt::IterateInitializer, [400](#)
- ~IteratesVector
 - Ipopt::IteratesVector, [404](#)
- ~IteratesVectorSpace
 - Ipopt::IteratesVectorSpace, [413](#)
- ~IterationOutput
 - Ipopt::IterationOutput, [416](#)
- ~IterativePardisoSolverInterface
 - Ipopt::IterativePardisoSolverInterface, [421](#)
- ~IterativeSolverTerminationTester
 - Ipopt::IterativeSolverTerminationTester, [429](#)
- ~IterativeWsmvSolverInterface
 - Ipopt::IterativeWsmvSolverInterface, [433](#)
- ~Journal
 - Ipopt::Journal, [438](#)
- ~Journalist
 - Ipopt::Journalist, [442](#)
- ~LeastSquareMultipliers
 - Ipopt::LeastSquareMultipliers, [445](#)
- ~LimMemQuasiNewtonUpdater
 - Ipopt::LimMemQuasiNewtonUpdater, [451](#)
- ~LineSearch
 - Ipopt::LineSearch, [460](#)
- ~LoqoMuOracle
 - Ipopt::LoqoMuOracle, [462](#)
- ~LowRankAugSystemSolver
 - Ipopt::LowRankAugSystemSolver, [466](#)
- ~LowRankSSAugSystemSolver
 - Ipopt::LowRankSSAugSystemSolver, [472](#)
- ~LowRankUpdateSymMatrix
 - Ipopt::LowRankUpdateSymMatrix, [478](#)
- ~LowRankUpdateSymMatrixSpace
 - Ipopt::LowRankUpdateSymMatrixSpace, [482](#)
- ~Ma27TSolverInterface
 - Ipopt::Ma27TSolverInterface, [487](#)
- ~Ma28TDependencyDetector
 - Ipopt::Ma28TDependencyDetector, [493](#)
- ~Ma57TSolverInterface
 - Ipopt::Ma57TSolverInterface, [496](#)
- ~Ma77SolverInterface
 - Ipopt::Ma77SolverInterface, [509](#)
- ~Ma86SolverInterface
 - Ipopt::Ma86SolverInterface, [518](#)
- ~Ma97SolverInterface
 - Ipopt::Ma97SolverInterface, [528](#)
- ~Matrix
 - Ipopt::Matrix, [535](#)
- ~MatrixSpace
 - Ipopt::MatrixSpace, [541](#)
- ~Mc19TSymScalingMethod
 - Ipopt::Mc19TSymScalingMethod, [543](#)
- ~MinC_1NrmRestorationPhase
 - Ipopt::MinC_1NrmRestorationPhase, [548](#)
- ~MonotoneMuUpdate
 - Ipopt::MonotoneMuUpdate, [551](#)
- ~MuOracle
 - Ipopt::MuOracle, [567](#)
- ~MuUpdate
 - Ipopt::MuUpdate, [569](#)
- ~MultiVectorMatrix
 - Ipopt::MultiVectorMatrix, [555](#)
- ~MultiVectorMatrixSpace

- Ipopt::MultiVectorMatrixSpace, 559
- ~MumpsSolverInterface
 - Ipopt::MumpsSolverInterface, 563
- ~NLP
 - Ipopt::NLP, 572
- ~NLPBoundsRemover
 - Ipopt::NLPBoundsRemover, 577
- ~NLPScalingObject
 - Ipopt::NLPScalingObject, 584
- ~NoNLPScalingObject
 - Ipopt::NoNLPScalingObject, 589
- ~Observer
 - Ipopt::Observer, 591
- ~OptimalityErrorConvergenceCheck
 - Ipopt::OptimalityErrorConvergenceCheck, 595
- ~OptionValue
 - Ipopt::OptionsList::OptionValue, 603
- ~OptionsList
 - Ipopt::OptionsList, 600
- ~OrigIpoptNLP
 - Ipopt::OrigIpoptNLP, 610
- ~OrigIterationOutput
 - Ipopt::OrigIterationOutput, 622
- ~PDFullSpaceSolver
 - Ipopt::PDFullSpaceSolver, 633
- ~PDPerturbationHandler
 - Ipopt::PDPerturbationHandler, 639
- ~PDSearchDirCalculator
 - Ipopt::PDSearchDirCalculator, 644
- ~PDSolver
 - Ipopt::PDSolver, 647
- ~PardisoSolverInterface
 - Ipopt::PardisoSolverInterface, 627
- ~PenaltyLSAcceptor
 - Ipopt::PenaltyLSAcceptor, 651
- ~PiecewisePenalty
 - Ipopt::PiecewisePenalty, 656
- ~PointPerturber
 - Ipopt::PointPerturber, 660
- ~ProbingMuOracle
 - Ipopt::ProbingMuOracle, 663
- ~QualityFunctionMuOracle
 - Ipopt::QualityFunctionMuOracle, 668
- ~ReferencedObject
 - Ipopt::ReferencedObject, 676
- ~RegisteredOption
 - Ipopt::RegisteredOption, 681
- ~RegisteredOptions
 - Ipopt::RegisteredOptions, 689
- ~RestoConvergenceCheck
 - Ipopt::RestoConvergenceCheck, 695
- ~RestoFilterConvergenceCheck
 - Ipopt::RestoFilterConvergenceCheck, 698
- ~RestoIpoptNLP
 - Ipopt::RestoIpoptNLP, 704
- ~RestoIterateInitializer
 - Ipopt::RestoIterateInitializer, 714
- ~RestoIterationOutput
 - Ipopt::RestoIterationOutput, 717
- ~RestoPenaltyConvergenceCheck
 - Ipopt::RestoPenaltyConvergenceCheck, 719
- ~RestoRestorationPhase
 - Ipopt::RestoRestorationPhase, 724
- ~RestorationPhase
 - Ipopt::RestorationPhase, 722
- ~ScaledMatrix
 - Ipopt::ScaledMatrix, 726
- ~ScaledMatrixSpace
 - Ipopt::ScaledMatrixSpace, 730
- ~SearchDirectionCalculator
 - Ipopt::SearchDirectionCalculator, 733
- ~SlackBasedTSymScalingMethod
 - Ipopt::SlackBasedTSymScalingMethod, 734
- ~SmartPtr
 - Ipopt::SmartPtr, 739
- ~SolveStatistics
 - Ipopt::SolveStatistics, 744
- ~SparseSymLinearSolverInterface
 - Ipopt::SparseSymLinearSolverInterface, 750
- ~StandardScalingBase
 - Ipopt::StandardScalingBase, 755
- ~StdAugSystemSolver
 - Ipopt::StdAugSystemSolver, 762
- ~StdInterfaceTNLP
 - Ipopt::StdInterfaceTNLP, 770
- ~StreamJournal
 - Ipopt::StreamJournal, 776
- ~Subject
 - Ipopt::Subject, 780
- ~SumMatrix
 - Ipopt::SumMatrix, 783
- ~SumMatrixSpace
 - Ipopt::SumMatrixSpace, 786
- ~SumSymMatrix
 - Ipopt::SumSymMatrix, 789
- ~SumSymMatrixSpace
 - Ipopt::SumSymMatrixSpace, 791
- ~SymLinearSolver
 - Ipopt::SymLinearSolver, 794
- ~SymMatrix
 - Ipopt::SymMatrix, 796
- ~SymMatrixSpace
 - Ipopt::SymMatrixSpace, 799
- ~SymScaledMatrix
 - Ipopt::SymScaledMatrix, 802
- ~SymScaledMatrixSpace
 - Ipopt::SymScaledMatrixSpace, 805
- ~SymTMatrix

- Ipopt::SymTMatrix, 809
- ~SymTMatrixSpace
 - Ipopt::SymTMatrixSpace, 812
- ~TDependencyDetector
 - Ipopt::TDependencyDetector, 818
- ~TNLP
 - Ipopt::TNLP, 832
- ~TNLPAdapter
 - Ipopt::TNLPAdapter, 842
- ~TNLPReducer
 - Ipopt::TNLPReducer, 854
- ~TSymDependencyDetector
 - Ipopt::TSymDependencyDetector, 876
- ~TSymLinearSolver
 - Ipopt::TSymLinearSolver, 880
- ~TSymScalingMethod
 - Ipopt::TSymScalingMethod, 884
- ~TaggedObject
 - Ipopt::TaggedObject, 816
- ~TimedTask
 - Ipopt::TimedTask, 820
- ~TimingStatistics
 - Ipopt::TimingStatistics, 824
- ~TransposeMatrix
 - Ipopt::TransposeMatrix, 860
- ~TransposeMatrixSpace
 - Ipopt::TransposeMatrixSpace, 863
- ~TripletEntry
 - Ipopt::TripletToCSRConverter::TripletEntry, 865
- ~TripletToCSRConverter
 - Ipopt::TripletToCSRConverter, 872
- ~UserScaling
 - Ipopt::UserScaling, 886
- ~Vector
 - Ipopt::Vector, 891
- ~VectorSpace
 - Ipopt::VectorSpace, 900
- ~WarmStartIterateInitializer
 - Ipopt::WarmStartIterateInitializer, 903
- ~WsmpSolverInterface
 - Ipopt::WsmpSolverInterface, 908
- ~ZeroMatrix
 - Ipopt::ZeroMatrix, 913
- ~ZeroMatrixSpace
 - Ipopt::ZeroMatrixSpace, 915
- ~ZeroSymMatrix
 - Ipopt::ZeroSymMatrix, 918
- ~ZeroSymMatrixSpace
 - Ipopt::ZeroSymMatrixSpace, 920

A

ADAPTIVE

- Ipopt::InexactSearchDirCalculator, 325

AFFINE_CORRECTOR

- Ipopt::FilterLSAcceptor, 250

ALL_VARS

- Ipopt, 46

ALWAYS

- Ipopt::InexactSearchDirCalculator, 325

a_

- Ipopt::IterativePardisoSolverInterface, 423

- Ipopt::IterativeWsmpSolverInterface, 435

- Ipopt::Ma27TSolverInterface, 491

- Ipopt::Ma57TSolverInterface, 500

- Ipopt::PardisoSolverInterface, 628

- Ipopt::WsmpSolverInterface, 910

ASSERT_EXCEPTION

- IpException.hpp, 969

accept_after_max_steps_

- Ipopt::BacktrackingLineSearch, 104

accept_every_trial_step_

- Ipopt::BacktrackingLineSearch, 104

AcceptTrialPoint

- Ipopt::CGPenaltyData, 126

- Ipopt::InexactData, 292

- Ipopt::IpoptAdditionalData, 332

- Ipopt::IpoptAlgorithm, 336

- Ipopt::IpoptData, 382

- Ipopt::TimingStatistics, 825

AcceptTrialPoint_

- Ipopt::TimingStatistics, 827

Acceptable

- Ipopt::Filter, 243, 244

- Ipopt::FilterEntry, 246

- Ipopt::PiecewisePenalty, 657

acceptable_compl_inf_tol_

- Ipopt::OptimalityErrorConvergenceCheck, 596

acceptable_constr_viol_tol_

- Ipopt::OptimalityErrorConvergenceCheck, 596

acceptable_counter_

- Ipopt::OptimalityErrorConvergenceCheck, 597

acceptable_dual_inf_tol_

- Ipopt::OptimalityErrorConvergenceCheck, 596

acceptable_iter_

- Ipopt::OptimalityErrorConvergenceCheck, 596

acceptable_iterate_

- Ipopt::BacktrackingLineSearch, 105

acceptable_iteration_number_

- Ipopt::BacktrackingLineSearch, 106

acceptable_obj_change_tol_

- Ipopt::OptimalityErrorConvergenceCheck, 597

acceptable_tol_

- Ipopt::OptimalityErrorConvergenceCheck, 596

accepted_by_Armijo_

- Ipopt::CGPenaltyLSAcceptor, 136

accepted_by_low_only_

- Ipopt::InexactLSAcceptor, 305

accepted_point_

- [Ipopt::AdaptiveMuUpdate](#), 63
- [acceptor_](#)
 - [Ipopt::BacktrackingLineSearch](#), 106
- [action](#)
 - [ma77_control_d](#), 502
 - [ma86_control_d](#), 513
 - [ma97_control_d](#), 521
- [ActivateFallbackMechanism](#)
 - [Ipopt::BacktrackingLineSearch](#), 101
 - [Ipopt::LineSearch](#), 461
- [adapt_to_target_mu](#)
 - [Ipopt::WarmStartIterateInitializer](#), 904
- [adaptive_mu_globalization_](#)
 - [Ipopt::AdaptiveMuUpdate](#), 61
- [adaptive_mu_kkt_balancing_term_](#)
 - [Ipopt::AdaptiveMuUpdate](#), 61
- [adaptive_mu_kkt_centrality_](#)
 - [Ipopt::AdaptiveMuUpdate](#), 61
- [adaptive_mu_kkt_norm_](#)
 - [Ipopt::AdaptiveMuUpdate](#), 61
- [adaptive_mu_monotone_init_factor_](#)
 - [Ipopt::AdaptiveMuUpdate](#), 61
- [adaptive_mu_safeguard_factor_](#)
 - [Ipopt::AdaptiveMuUpdate](#), 61
- [AdaptiveMuGlobalizationEnum](#)
 - [Ipopt::AdaptiveMuUpdate](#), 58
- [AdaptiveMuUpdate](#)
 - [Ipopt::AdaptiveMuUpdate](#), 59
- [add_cq_](#)
 - [Ipopt::IpoptCalculatedQuantities](#), 365
- [add_data_](#)
 - [Ipopt::IpoptData](#), 388
- [AddAmplOption](#)
 - [Ipopt::AmplOptionsList](#), 74
- [AddAvailableSuffix](#)
 - [Ipopt::AmplSuffixHandler](#), 76
- [AddBoundedIntegerOption](#)
 - [Ipopt::RegisteredOptions](#), 691
- [AddBoundedNumberOption](#)
 - [Ipopt::RegisteredOptions](#), 690
- [AddCachedResult](#)
 - [Ipopt::CachedResults](#), 114
- [AddCachedResult1Dep](#)
 - [Ipopt::CachedResults](#), 115, 116
- [AddCachedResult2Dep](#)
 - [Ipopt::CachedResults](#), 115, 116
- [AddCachedResult3Dep](#)
 - [Ipopt::CachedResults](#), 115, 116
- [AddEntry](#)
 - [Ipopt::Filter](#), 244
 - [Ipopt::PiecewisePenalty](#), 657
- [AddFileJournal](#)
 - [Ipopt::Journalist](#), 443
- [AddInexactDefaultOptions](#)
 - [Ipopt](#), 49
- [AddIntegerOption](#)
 - [Ipopt::RegisteredOptions](#), 690
- [AddJournal](#)
 - [Ipopt::Journalist](#), 443
- [AddLowerBoundedIntegerOption](#)
 - [Ipopt::RegisteredOptions](#), 690
- [AddLowerBoundedNumberOption](#)
 - [Ipopt::RegisteredOptions](#), 690
- [AddMSinvZ](#)
 - [Ipopt::Matrix](#), 536
- [AddMSinvZImpl](#)
 - [Ipopt::CompoundMatrix](#), 154
 - [Ipopt::ExpansionMatrix](#), 236
 - [Ipopt::IdentityMatrix](#), 279
 - [Ipopt::Matrix](#), 537
 - [Ipopt::ScaledMatrix](#), 728
- [AddMatrix](#)
 - [Ipopt::DenseSymMatrix](#), 198
- [AddMatrixProduct](#)
 - [Ipopt::DenseGenMatrix](#), 191
- [AddNumberOption](#)
 - [Ipopt::RegisteredOptions](#), 690
- [AddOneMultiVectorMatrix](#)
 - [Ipopt::MultiVectorMatrix](#), 556
- [AddOneVector](#)
 - [Ipopt::Vector](#), 894
- [AddRef](#)
 - [Ipopt::ReferencedObject](#), 676
- [AddRightMultMatrix](#)
 - [Ipopt::MultiVectorMatrix](#), 556
- [AddScalar](#)
 - [Ipopt::Vector](#), 893
- [AddScalarImpl](#)
 - [Ipopt::CompoundVector](#), 174
 - [Ipopt::DenseVector](#), 207
 - [Ipopt::Vector](#), 896
- [AddStringOption](#)
 - [Ipopt::RegisteredOptions](#), 691
- [AddStringOption1](#)
 - [Ipopt::RegisteredOptions](#), 691
- [AddStringOption10](#)
 - [Ipopt::RegisteredOptions](#), 692
- [AddStringOption2](#)
 - [Ipopt::RegisteredOptions](#), 691
- [AddStringOption3](#)
 - [Ipopt::RegisteredOptions](#), 691
- [AddStringOption4](#)
 - [Ipopt::RegisteredOptions](#), 691
- [AddStringOption5](#)
 - [Ipopt::RegisteredOptions](#), 691
- [AddStringOption6](#)
 - [Ipopt::RegisteredOptions](#), 691
- [AddStringOption7](#)

- Ipopt::RegisteredOptions, 691
- AddStringOption8
 - Ipopt::RegisteredOptions, 692
- AddStringOption9
 - Ipopt::RegisteredOptions, 692
- AddTwoVectors
 - Ipopt::Vector, 894
- AddTwoVectorsImpl
 - Ipopt::CompoundVector, 174
 - Ipopt::DenseVector, 207
 - Ipopt::Vector, 897
- AddUpperBoundedIntegerOption
 - Ipopt::RegisteredOptions, 690
- AddUpperBoundedNumberOption
 - Ipopt::RegisteredOptions, 690
- AddValidStringSetting
 - Ipopt::RegisteredOption, 683
- AddVectorQuotient
 - Ipopt::Vector, 894
- AddVectorQuotientImpl
 - Ipopt::CompoundVector, 175
 - Ipopt::DenseVector, 208
 - Ipopt::Vector, 897
- AdditionalCq
 - Ipopt::IpoptCalculatedQuantities, 364
- AdditionalData
 - Ipopt::IpoptData, 385
- AdjustVariableBounds
 - Ipopt::IpoptNLP, 397
 - Ipopt::OrigIpoptNLP, 613
 - Ipopt::RestIpoptNLP, 707
- AdjustedTrialSlacks
 - Ipopt::IpoptCalculatedQuantities, 356
- airn_
 - Ipopt::TSymLinearSolver, 883
- ajcn_
 - Ipopt::TSymLinearSolver, 883
- akeep_
 - Ipopt::Ma97SolverInterface, 531
- alg_
 - Ipopt::IpoptApplication, 344
- Algorithm/Inexact/IpInexactAlgBuilder.hpp, 921
- Algorithm/Inexact/IpInexactCq.hpp, 921
- Algorithm/Inexact/IpInexactData.hpp, 922
- Algorithm/Inexact/IpInexactDoglegNormal.hpp, 922
- Algorithm/Inexact/IpInexactLSAcceptor.hpp, 922
- Algorithm/Inexact/IpInexactNewtonNormal.hpp, 923
- Algorithm/Inexact/IpInexactNormalStepCalc.hpp, 923
- Algorithm/Inexact/IpInexactNormalTerminationTester.hpp, 923
- Algorithm/Inexact/IpInexactPDSolver.hpp, 924
- Algorithm/Inexact/IpInexactPDTerminationTester.hpp, 924
- Algorithm/Inexact/IpInexactRegOp.hpp, 924
- Algorithm/Inexact/IpInexactSearchDirCalc.hpp, 925
- Algorithm/Inexact/IpInexactTSymScalingMethod.hpp, 925
- Algorithm/Inexact/IpIterativePardisoSolverInterface.hpp, 925
- Algorithm/Inexact/IpIterativeSolverTerminationTester.hpp, 926
- Algorithm/IpAdaptiveMuUpdate.hpp, 926
- Algorithm/IpAlgBuilder.hpp, 926
- Algorithm/IpAlgStrategy.hpp, 927
- Algorithm/IpAlgorithmRegOp.hpp, 927
- Algorithm/IpAugRestoSystemSolver.hpp, 927
- Algorithm/IpAugSystemSolver.hpp, 928
- Algorithm/IpBacktrackingLSAcceptor.hpp, 929
- Algorithm/IpBacktrackingLineSearch.hpp, 928
- Algorithm/IpConvCheck.hpp, 929
- Algorithm/IpDefaultIterateInitializer.hpp, 929
- Algorithm/IpEqMultCalculator.hpp, 930
- Algorithm/IpEquilibrationScaling.hpp, 930
- Algorithm/IpExactHessianUpdater.hpp, 930
- Algorithm/IpFilter.hpp, 931
- Algorithm/IpFilterLSAcceptor.hpp, 931
- Algorithm/IpGenAugSystemSolver.hpp, 931
- Algorithm/IpGradientScaling.hpp, 932
- Algorithm/IpHessianUpdater.hpp, 932
- Algorithm/IpIpoptAlg.hpp, 932
- Algorithm/IpIpoptCalculatedQuantities.hpp, 933
- Algorithm/IpIpoptData.hpp, 933
- Algorithm/IpIpoptNLP.hpp, 934
- Algorithm/IpIterateInitializer.hpp, 934
- Algorithm/IpIteratesVector.hpp, 935
- Algorithm/IpIterationOutput.hpp, 935
- Algorithm/IpLeastSquareMults.hpp, 935
- Algorithm/IpLimMemQuasiNewtonUpdater.hpp, 936
- Algorithm/IpLineSearch.hpp, 936
- Algorithm/IpLoqoMuOracle.hpp, 936
- Algorithm/IpLowRankAugSystemSolver.hpp, 937
- Algorithm/IpLowRankSSAugSystemSolver.hpp, 937
- Algorithm/IpMonotoneMuUpdate.hpp, 937
- Algorithm/IpMuOracle.hpp, 938
- Algorithm/IpMuUpdate.hpp, 938
- Algorithm/IpNLPBoundsRemover.hpp, 938
- Algorithm/IpNLPScaling.hpp, 939
- Algorithm/IpOptErrorConvCheck.hpp, 939
- Algorithm/IpOrigIpoptNLP.hpp, 939
- Algorithm/IpOrigIterationOutput.hpp, 940
- Algorithm/IpPDFullSpaceSolver.hpp, 940
- Algorithm/IpPDPerturbationHandler.hpp, 940
- Algorithm/IpPDSearchDirCalc.hpp, 941
- Algorithm/IpPDSolver.hpp, 941
- Algorithm/IpPenaltyLSAcceptor.hpp, 941
- Algorithm/IpProbingMuOracle.hpp, 942
- Algorithm/IpQualityFunctionMuOracle.hpp, 942
- Algorithm/IpRestoConvCheck.hpp, 942
- Algorithm/IpRestoFilterConvCheck.hpp, 943
- Algorithm/IpRestIpoptNLP.hpp, 943

- Algorithm/IpRestoIterateInitializer.hpp, 943
- Algorithm/IpRestoIterationOutput.hpp, 944
- Algorithm/IpRestoMinC_1Nrm.hpp, 944
- Algorithm/IpRestoPenaltyConvCheck.hpp, 944
- Algorithm/IpRestoPhase.hpp, 945
- Algorithm/IpRestoRestoPhase.hpp, 945
- Algorithm/IpSearchDirCalculator.hpp, 946
- Algorithm/IpStdAugSystemSolver.hpp, 946
- Algorithm/IpTimingStatistics.hpp, 946
- Algorithm/IpUserScaling.hpp, 947
- Algorithm/IpWarmStartIterateInitializer.hpp, 947
- Algorithm/LinearSolvers/IpGenKKTSolverInterface.hpp, 958
- Algorithm/LinearSolvers/IpIterativeWsmvSolverInterface.-hpp, 958
- Algorithm/LinearSolvers/IpLinearSolversRegOp.hpp, 958
- Algorithm/LinearSolvers/IpMa27TSolverInterface.hpp, 959
- Algorithm/LinearSolvers/IpMa28TDependencyDetector.-hpp, 959
- Algorithm/LinearSolvers/IpMa57TSolverInterface.hpp, 959
- Algorithm/LinearSolvers/IpMa77SolverInterface.hpp, 960
- Algorithm/LinearSolvers/IpMa86SolverInterface.hpp, 960
- Algorithm/LinearSolvers/IpMa97SolverInterface.hpp, 960
- Algorithm/LinearSolvers/IpMc19TSymScalingMethod.hpp, 961
- Algorithm/LinearSolvers/IpMumpsSolverInterface.hpp, 961
- Algorithm/LinearSolvers/IpPardisoSolverInterface.hpp, 961
- Algorithm/LinearSolvers/IpSlackBasedTSymScaling-Method.hpp, 962
- Algorithm/LinearSolvers/IpSparseSymLinearSolver-Interface.hpp, 962
- Algorithm/LinearSolvers/IpSymLinearSolver.hpp, 962
- Algorithm/LinearSolvers/IpTDependencyDetector.hpp, 963
- Algorithm/LinearSolvers/IpTSymDependencyDetector.-hpp, 964
- Algorithm/LinearSolvers/IpTSymLinearSolver.hpp, 964
- Algorithm/LinearSolvers/IpTSymScalingMethod.hpp, 964
- Algorithm/LinearSolvers/IpTripletToCSRConverter.hpp, 963
- Algorithm/LinearSolvers/IpWsmvSolverInterface.hpp, 965
- Algorithm/LinearSolvers/hsl_ma77d.h, 947
- Algorithm/LinearSolvers/hsl_ma86d.h, 951
- Algorithm/LinearSolvers/hsl_ma97d.h, 953
- Algorithm/LinearSolvers/hsl_mc68i.h, 957
- AlgorithmBuilder
 - Ipopt::AlgorithmBuilder, 64
- AlgorithmMode
 - Ipopt, 49
 - IpReturnCodes_inc.h, 998
- AlgorithmObject
 - Ipopt::IpoptApplication, 343
- AlgorithmStrategyObject
 - Ipopt::AlgorithmStrategyObject, 67, 68
- allocate_block_
 - Ipopt::CompoundMatrixSpace, 160
 - Ipopt::CompoundSymMatrixSpace, 168
- AllocateInternalStorage
 - Ipopt::DenseVectorSpace, 211
 - Ipopt::GenTMatrixSpace, 271
 - Ipopt::SymTMatrixSpace, 813
- allow_clobber_
 - Ipopt::OptionsList::OptionValue, 604
- allow_twosided_inequalities_
 - Ipopt::NLPBoundsRemover, 581
- AllowClobber
 - Ipopt::OptionsList::OptionValue, 604
- alpha_for_y_
 - Ipopt::BacktrackingLineSearch, 103
- alpha_for_y_tol_
 - Ipopt::BacktrackingLineSearch, 103
- alpha_min_frac_
 - Ipopt::FilterLSAcceptor, 254
- alpha_red_factor_
 - Ipopt::BacktrackingLineSearch, 103
- AlphaForYEnum
 - Ipopt::BacktrackingLineSearch, 100
- Amax
 - Ipopt::Vector, 892
- amax_cache_tag_
 - Ipopt::Vector, 898
- AmaxImpl
 - Ipopt::CompoundVector, 173
 - Ipopt::DenseVector, 206
 - Ipopt::Vector, 895
- ampl_options_map_
 - Ipopt::AmplOptionsList, 74
- AmplOption
 - Ipopt::AmplOptionsList::AmplOption, 71
- AmplOptionType
 - Ipopt::AmplOptionsList, 73
- AmplOptionsList
 - Ipopt::AmplOptionsList, 73
- AmplSolverObject
 - Ipopt::AmplTNLP, 83
- AmplSuffixHandler
 - Ipopt::AmplSuffixHandler, 76
- AmplTNLP
 - Ipopt::AmplSuffixHandler, 77
 - Ipopt::AmplTNLP, 81
- Append_info_string
 - Ipopt::IpoptData, 384
- ApplicationReturnStatus
 - Ipopt, 48
 - IpReturnCodes_inc.h, 997
- apply_grad_obj_scaling

- Ipopt::NLPScalingObject, [587](#)
- apply_grad_obj_scaling_NonConst
 - Ipopt::NLPScalingObject, [586](#)
- apply_hessian_scaling
 - Ipopt::NLPScalingObject, [586](#)
 - Ipopt::StandardScalingBase, [757](#)
- apply_jac_c_scaling
 - Ipopt::NLPScalingObject, [586](#)
 - Ipopt::StandardScalingBase, [757](#)
- apply_jac_d_scaling
 - Ipopt::NLPScalingObject, [586](#)
 - Ipopt::StandardScalingBase, [757](#)
- apply_new_x
 - Ipopt::AmplTNLP, [85](#)
 - Ipopt::StdInterfaceTNLP, [772](#)
- apply_obj_scaling
 - Ipopt::NLPScalingObject, [584](#)
 - Ipopt::StandardScalingBase, [756](#)
- apply_vector_scaling_c
 - Ipopt::NLPScalingObject, [585](#)
 - Ipopt::StandardScalingBase, [756](#)
- apply_vector_scaling_c_NonConst
 - Ipopt::NLPScalingObject, [585](#)
 - Ipopt::StandardScalingBase, [756](#)
- apply_vector_scaling_d
 - Ipopt::NLPScalingObject, [585](#)
 - Ipopt::StandardScalingBase, [757](#)
- apply_vector_scaling_d_LU
 - Ipopt::NLPScalingObject, [586](#)
- apply_vector_scaling_d_LU_NonConst
 - Ipopt::NLPScalingObject, [586](#)
- apply_vector_scaling_d_NonConst
 - Ipopt::NLPScalingObject, [585](#)
 - Ipopt::StandardScalingBase, [757](#)
- apply_vector_scaling_x
 - Ipopt::NLPScalingObject, [584](#)
 - Ipopt::StandardScalingBase, [756](#)
- apply_vector_scaling_x_LU
 - Ipopt::NLPScalingObject, [586](#)
- apply_vector_scaling_x_LU_NonConst
 - Ipopt::NLPScalingObject, [586](#)
- apply_vector_scaling_x_NonConst
 - Ipopt::NLPScalingObject, [584](#)
 - Ipopt::StandardScalingBase, [756](#)
- Apps/AmplSolver/AmplTNLP.hpp, [965](#)
- ArmijoHolds
 - Ipopt::CGPenaltyLSAceptor, [134](#)
 - Ipopt::FilterLSAceptor, [252](#)
- asl_
 - Ipopt::AmplSuffixHandler, [77](#)
 - Ipopt::AmplTNLP, [85](#)
- Asum
 - Ipopt::Vector, [892](#)
- asum_cache_tag_
 - Ipopt::Vector, [898](#)
- AsumImpl
 - Ipopt::CompoundVector, [173](#)
 - Ipopt::DenseVector, [206](#)
 - Ipopt::Vector, [895](#)
- atag_
 - Ipopt::TSymLinearSolver, [881](#)
- AttachObserver
 - Ipopt::Subject, [780](#)
- aug_solver_
 - Ipopt::InexactNewtonNormalStep, [307](#)
- aug_system_solver_
 - Ipopt::DefaultIterateInitializer, [187](#)
 - Ipopt::LowRankAugSystemSolver, [467](#)
 - Ipopt::LowRankSSAugSystemSolver, [474](#)
- AugRestoSystemSolver
 - Ipopt::AugRestoSystemSolver, [89, 90](#)
- augSysSolver_
 - Ipopt::InexactPDSolver, [316](#)
 - Ipopt::PDFullSpaceSolver, [634](#)
- AugSystemSolver
 - Ipopt::AugSystemSolver, [94](#)
- AugmentDenseVector
 - Ipopt::LimMemQuasiNewtonUpdater, [452](#)
- AugmentFilter
 - Ipopt::FilterLSAceptor, [253](#)
- AugmentLMatrix
 - Ipopt::LimMemQuasiNewtonUpdater, [452](#)
- AugmentMultiVector
 - Ipopt::LimMemQuasiNewtonUpdater, [452](#)
- AugmentSTDRSMatrix
 - Ipopt::LimMemQuasiNewtonUpdater, [452](#)
- AugmentSdotSMatrix
 - Ipopt::LimMemQuasiNewtonUpdater, [452](#)
- augmented_system_
 - Ipopt::StdAugSystemSolver, [766](#)
- augmented_system_space_
 - Ipopt::StdAugSystemSolver, [764](#)
- augmented_vector_space_
 - Ipopt::StdAugSystemSolver, [764](#)
- AugmentedSystemChanged
 - Ipopt::GenAugSystemSolver, [259](#)
- AugmentedSystemRequiresChange
 - Ipopt::LowRankAugSystemSolver, [467](#)
 - Ipopt::LowRankSSAugSystemSolver, [474](#)
 - Ipopt::StdAugSystemSolver, [763](#)
- augsys_improved_
 - Ipopt::PDFullSpaceSolver, [634](#)
- augsys_tag_
 - Ipopt::StdAugSystemSolver, [766](#)
- augsysolver_
 - Ipopt::LeastSquareMultipliers, [445](#)
- Axpy
 - Ipopt::Vector, [892](#)

- AxpyImpl
 - Ipopt::CompoundVector, [173](#)
 - Ipopt::DenseVector, [205](#)
 - Ipopt::Vector, [895](#)
- B
- B0_
 - Ipopt::LimMemQuasiNewtonUpdater, [456](#)
- B0_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [458](#)
- B_CONSTANT
 - Ipopt::DefaultIterateInitializer, [185](#)
- B_MU_BASED
 - Ipopt::DefaultIterateInitializer, [185](#)
- BFGS
 - Ipopt::LimMemQuasiNewtonUpdater, [450](#)
- BT_CUBIC
 - Ipopt::QualityFunctionMuOracle, [668](#)
- BT_NONE
 - Ipopt::QualityFunctionMuOracle, [668](#)
- Backsolve
 - Ipopt::Ma27TSolverInterface, [488](#)
 - Ipopt::Ma57TSolverInterface, [498](#)
- BacktrackingLSAceptor
 - Ipopt::BacktrackingLSAceptor, [108](#), [109](#)
- BacktrackingLineSearch
 - Ipopt::BacktrackingLineSearch, [100](#)
- BalancingTermEnum
 - Ipopt::QualityFunctionMuOracle, [668](#)
- barrier_obj
 - Ipopt::PiecewisePenEntry, [658](#)
- barrier_tol_factor_
 - Ipopt::AdaptiveMuUpdate, [61](#)
 - Ipopt::MonotoneMuUpdate, [552](#)
- best_KKT_error_
 - Ipopt::CGPenaltyLSAceptor, [137](#)
- best_iterate_
 - Ipopt::CGPenaltyLSAceptor, [137](#)
- BiggestBarr
 - Ipopt::PiecewisePenalty, [657](#)
- bits
 - ma77_control_d, [501](#)
- block_cols_
 - Ipopt::CompoundMatrixSpace, [160](#)
- block_dim_
 - Ipopt::CompoundSymMatrixSpace, [168](#)
- block_rows_
 - Ipopt::CompoundMatrixSpace, [160](#)
- Bool
 - IpStdCInterface.h, [1002](#)
- bound_frac_
 - Ipopt::DefaultIterateInitializer, [186](#)
- bound_mult_init_method_
 - Ipopt::DefaultIterateInitializer, [187](#)
- bound_mult_init_val_
 - Ipopt::DefaultIterateInitializer, [187](#)
- bound_mult_reset_threshold_
 - Ipopt::MinC_1NrmRestorationPhase, [549](#)
- bound_push_
 - Ipopt::DefaultIterateInitializer, [186](#)
- bound_relax_factor_
 - Ipopt::OrigIpoptNLP, [619](#)
 - Ipopt::TNLPAdapter, [846](#)
- BoundMultInitMethod
 - Ipopt::DefaultIterateInitializer, [185](#)
- buffer_
 - Ipopt::StreamJournal, [777](#)
- buffer_lpage
 - ma77_control_d, [501](#)
- buffer_npage
 - ma77_control_d, [502](#)
- BuildBasicAlgorithm
 - Ipopt::AlgorithmBuilder, [65](#)
 - Ipopt::InexactAlgorithmBuilder, [283](#)
- BuildIpoptObjects
 - Ipopt::AlgorithmBuilder, [65](#)
 - Ipopt::InexactAlgorithmBuilder, [283](#)
- C
- c
 - Ipopt::IpoptNLP, [395](#)
 - Ipopt::OrigIpoptNLP, [611](#)
 - Ipopt::RestIpoptNLP, [705](#)
- C_STYLE
 - Ipopt::TNLP, [832](#)
- CEN_CUBED_RECIPROCAL
 - Ipopt::QualityFunctionMuOracle, [668](#)
- CEN_LOG
 - Ipopt::QualityFunctionMuOracle, [668](#)
- CEN_NONE
 - Ipopt::QualityFunctionMuOracle, [668](#)
- CEN_RECIPROCAL
 - Ipopt::QualityFunctionMuOracle, [668](#)
- CHOL
 - Ipopt::DenseGenMatrix, [190](#)
- COMPLETE
 - Ipopt::IterativePardisoSolverInterface, [421](#)
 - Ipopt::PardisoSolverInterface, [626](#)
- COMPLETE2x2
 - Ipopt::IterativePardisoSolverInterface, [421](#)
 - Ipopt::PardisoSolverInterface, [626](#)
- CONSTANT
 - Ipopt::LimMemQuasiNewtonUpdater, [451](#)
- CONSTRAINT
 - Ipopt::IterativePardisoSolverInterface, [421](#)
 - Ipopt::PardisoSolverInterface, [627](#)
- CONTINUE
 - Ipopt::ConvergenceCheck, [181](#)

- Ipopt::IterativeSolverTerminationTester, [429](#)
- CONVERGED
 - Ipopt::ConvergenceCheck, [181](#)
- CONVERGED_TO_ACCEPTABLE_POINT
 - Ipopt::ConvergenceCheck, [181](#)
- CPUTIME_EXCEEDED
 - Ipopt, [48](#)
 - Ipopt::ConvergenceCheck, [181](#)
- CSR_Format_0_Offset
 - Ipopt::SparseSymLinearSolverInterface, [750](#)
- CSR_Format_1_Offset
 - Ipopt::SparseSymLinearSolverInterface, [750](#)
- CSR_Full_Format_0_Offset
 - Ipopt::SparseSymLinearSolverInterface, [750](#)
- CSR_Full_Format_1_Offset
 - Ipopt::SparseSymLinearSolverInterface, [750](#)
- c_Avc_norm_cauchy_
 - Ipopt::InexactNormalTerminationTester, [313](#)
- c_cache_
 - Ipopt::OrigIpoptNLP, [617](#)
- c_eval_time
 - Ipopt::OrigIpoptNLP, [615](#)
- c_eval_time_
 - Ipopt::OrigIpoptNLP, [620](#)
- c_evals
 - Ipopt::IpoptNLP, [397](#)
 - Ipopt::OrigIpoptNLP, [614](#)
 - Ipopt::RestIpoptNLP, [708](#)
- c_evals_
 - Ipopt::OrigIpoptNLP, [620](#)
 - Ipopt::RestIpoptNLP, [712](#)
- c_norm_
 - Ipopt::InexactPDTerminationTester, [322](#)
- c_plus_Av_norm_
 - Ipopt::InexactPDTerminationTester, [322](#)
- c_rhs_
 - Ipopt::TNLPAdapter, [849](#)
- c_space_
 - Ipopt::OrigIpoptNLP, [616](#)
 - Ipopt::RestIpoptNLP, [709](#)
 - Ipopt::TNLPAdapter, [848](#)
- CGPenCq
 - Ipopt::CGPenaltyLSAceptor, [134](#)
 - Ipopt::CGPerturbationHandler, [143](#)
 - Ipopt::CGSearchDirCalculator, [149](#)
- CGPenData
 - Ipopt::CGPenaltyCq, [120](#)
 - Ipopt::CGPenaltyLSAceptor, [134](#)
 - Ipopt::CGPerturbationHandler, [143](#)
 - Ipopt::CGSearchDirCalculator, [149](#)
- CGPenaltyCq
 - Ipopt::CGPenaltyCq, [119](#)
- CGPenaltyData
 - Ipopt::CGPenaltyData, [124](#)
- CGPenaltyLSAceptor
 - Ipopt::CGPenaltyLSAceptor, [132](#)
- CGPerturbationHandler
 - Ipopt::CGPerturbationHandler, [142](#)
- CGSearchDirCalculator
 - Ipopt::CGSearchDirCalculator, [148](#)
- COIN_HAS_AS_
 - config_default.h, [966](#)
- COIN_HAS_BLAS
 - config_default.h, [966](#)
- COIN_HAS_HSL
 - config_default.h, [966](#)
- COIN_HAS_LAPACK
 - config_default.h, [966](#)
- cache_priority_
 - Ipopt::TaggedObject, [817](#)
- cached_amax_
 - Ipopt::Vector, [898](#)
- cached_asum_
 - Ipopt::Vector, [898](#)
- cached_max_
 - Ipopt::Vector, [898](#)
- cached_min_
 - Ipopt::Vector, [898](#)
- cached_nrm2_
 - Ipopt::Vector, [898](#)
- cached_results_
 - Ipopt::CachedResults, [117](#)
- cached_sum_
 - Ipopt::Vector, [899](#)
- cached_sumlogs_
 - Ipopt::Vector, [899](#)
- cached_valid_
 - Ipopt::Matrix, [539](#)
 - Ipopt::Vector, [899](#)
- CachedResults
 - Ipopt::CachedResults, [114](#)
- calc_number_of_bounds
 - Ipopt::IpoptAlgorithm, [336](#)
- CalcBarrierTerm
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- CalcCentralityMeasure
 - Ipopt::IpoptCalculatedQuantities, [362](#)
- CalcCompl
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- CalcFracToBound
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- CalcNewMuAndTau
 - Ipopt::MonotoneMuUpdate, [552](#)
- CalcNormOfType
 - Ipopt::IpoptCalculatedQuantities, [363](#)
- CalcPred
 - Ipopt::InexactLSAceptor, [302](#)
 - Ipopt::PenaltyLSAceptor, [653](#)

- CalcSlack_L
 - Ipopt::IpoptCalculatedQuantities, 364
- CalcSlack_U
 - Ipopt::IpoptCalculatedQuantities, 364
- CalculateAffineMu
 - Ipopt::ProbingMuOracle, 664
- CalculateAlphaMin
 - Ipopt::BacktrackingLSAcceptor, 109
 - Ipopt::CGPenaltyLSAcceptor, 133
 - Ipopt::FilterLSAcceptor, 251
 - Ipopt::InexactLSAcceptor, 301
 - Ipopt::PenaltyLSAcceptor, 651
- CalculateLeastSquareDuals
 - Ipopt::DefaultIterateInitializer, 186
- CalculateLeastSquarePrimals
 - Ipopt::DefaultIterateInitializer, 186
- CalculateMu
 - Ipopt::LoqpMuOracle, 463
 - Ipopt::MuOracle, 568
 - Ipopt::ProbingMuOracle, 664
 - Ipopt::QualityFunctionMuOracle, 669
- CalculateMultipliers
 - Ipopt::EqMultiplierCalculator, 223
 - Ipopt::LeastSquareMultipliers, 445
- CalculateQualityFunction
 - Ipopt::QualityFunctionMuOracle, 669
- CalculateSafeSlack
 - Ipopt::IpoptCalculatedQuantities, 365
- call_heset
 - Ipopt::AmplTNLP, 85
- call_optimize
 - Ipopt::IpoptApplication, 344
- CentralityEnum
 - Ipopt::QualityFunctionMuOracle, 668
- check_derivatives_for_naninf_
 - Ipopt::OrigIpoptNLP, 619
- check_if_no_bounds_
 - Ipopt::AdaptiveMuUpdate, 63
- CheckAcceptabilityOfTrialPoint
 - Ipopt::BacktrackingLineSearch, 102
 - Ipopt::BacktrackingLSAcceptor, 109
 - Ipopt::CGPenaltyLSAcceptor, 133
 - Ipopt::FilterLSAcceptor, 251
 - Ipopt::InexactLSAcceptor, 301
 - Ipopt::PenaltyLSAcceptor, 651
- CheckConvergence
 - Ipopt::ConvergenceCheck, 182
 - Ipopt::OptimalityErrorConvergenceCheck, 595
 - Ipopt::RestoConvergenceCheck, 695
 - Ipopt::TimingStatistics, 825
- CheckConvergence_
 - Ipopt::TimingStatistics, 827
- CheckDerivatives
 - Ipopt::TNLPAdapter, 844
- CheckSkippedLineSearch
 - Ipopt::BacktrackingLineSearch, 101
 - Ipopt::LineSearch, 461
- CheckSkippingBFGS
 - Ipopt::LimMemQuasiNewtonUpdater, 451
- CheckSufficientProgress
 - Ipopt::AdaptiveMuUpdate, 59
- chi_cup_
 - Ipopt::CGPenaltyLSAcceptor, 135
- chi_hat_
 - Ipopt::CGPenaltyLSAcceptor, 135
- chi_tilde_
 - Ipopt::CGPenaltyLSAcceptor, 135
- CholeskyBackSolveMatrix
 - Ipopt::DenseGenMatrix, 192
- CholeskySolveMatrix
 - Ipopt::DenseGenMatrix, 192
- CholeskySolveVector
 - Ipopt::DenseGenMatrix, 192
- CleanupInvalidatedResults
 - Ipopt::CachedResults, 116
- Clear
 - Ipopt::CachedResults, 116
 - Ipopt::Filter, 244
 - Ipopt::InexactNormalTerminationTester, 312
 - Ipopt::InexactPDTerminationTester, 320
 - Ipopt::IterativeSolverTerminationTester, 430
 - Ipopt::PiecewisePenalty, 657
- clear
 - Ipopt::OptionsList, 600
- clone
 - Ipopt::IpoptApplication, 341
- cntl_
 - Ipopt::Ma27TSolverInterface, 490
- ColVectorSpace
 - Ipopt::MultiVectorMatrix, 557
 - Ipopt::MultiVectorMatrixSpace, 560
- column_scaling_
 - Ipopt::ScaledMatrixSpace, 731
- ColumnScaling
 - Ipopt::ScaledMatrix, 727
 - Ipopt::ScaledMatrixSpace, 731
- Common/IpCachedResults.hpp, 967
- Common/IpDebug.hpp, 967
- Common/IpException.hpp, 969
- Common/IpJournalist.hpp, 970
- Common/IpObserver.hpp, 971
- Common/IpOptionsList.hpp, 971
- Common/IpReferenced.hpp, 972
- Common/IpRegOptions.hpp, 972
- Common/IpSmartPtr.hpp, 973
- Common/IpTaggedObject.hpp, 974
- Common/IpTimedTask.hpp, 975
- Common/IpTypes.hpp, 975

- Common/IpUtils.hpp, 976
- Common/IpoptConfig.h, 971
- Common/config_default.h, 966
- Common/config_ipopt_default.h, 966
- Comp
 - Ipopt::CompoundMatrix, 155
 - Ipopt::CompoundSymMatrix, 164
 - Ipopt::CompoundVector, 175
- comp_spaces_
 - Ipopt::CompoundMatrixSpace, 160
 - Ipopt::CompoundSymMatrixSpace, 168
 - Ipopt::CompoundVectorSpace, 179
- Compare_le
 - Ipopt, 52
 - Ipopt::CGPenaltyLSAceptor, 135
- ComparePointers
 - Ipopt, 51
- compl_
 - Ipopt::SolveStatistics, 746
- compl_inf_tol_
 - Ipopt::AdaptiveMuUpdate, 61
 - Ipopt::MonotoneMuUpdate, 553
 - Ipopt::OptimalityErrorConvergenceCheck, 596
- compound_sol_vecspace_
 - Ipopt::LowRankAugSystemSolver, 469
- CompoundMatrix
 - Ipopt::CompoundMatrix, 153
- CompoundMatrixSpace
 - Ipopt::CompoundMatrixSpace, 158
- CompoundSymMatrix
 - Ipopt::CompoundSymMatrix, 162, 163
- CompoundSymMatrixSpace
 - Ipopt::CompoundSymMatrixSpace, 167
- CompoundVector
 - Ipopt::CompoundVector, 171, 172
- CompoundVectorSpace
 - Ipopt::CompoundVectorSpace, 177
- compressed_pos_
 - Ipopt::ExpansionMatrixSpace, 239
- CompressedPosIndices
 - Ipopt::ExpansionMatrix, 236
 - Ipopt::ExpansionMatrixSpace, 239
- comps_
 - Ipopt::CompoundMatrix, 155
 - Ipopt::CompoundSymMatrix, 164
 - Ipopt::CompoundVector, 175
- compute_curr_cg_penalty
 - Ipopt::CGPenaltyCq, 120
- compute_curr_cg_penalty_scale
 - Ipopt::CGPenaltyCq, 120
- compute_normal
 - Ipopt::InexactData, 293
- compute_normal_
 - Ipopt::InexactData, 293
- Compute_tau_monotone
 - Ipopt::AdaptiveMuUpdate, 60
- ComputeAcceptableTrialPoint
 - Ipopt::IpoptAlgorithm, 336
 - Ipopt::TimingStatistics, 825
- ComputeAcceptableTrialPoint_
 - Ipopt::TimingStatistics, 827
- ComputeAlphaForY
 - Ipopt::BacktrackingLSAceptor, 111
 - Ipopt::InexactLSAceptor, 302
- ComputeBoundMultiplierStep
 - Ipopt::MinC_1NrmRestorationPhase, 548
- ComputeCholeskyFactor
 - Ipopt::DenseGenMatrix, 191
- ComputeColAMax
 - Ipopt::Matrix, 536
- ComputeColAMaxImpl
 - Ipopt::CompoundMatrix, 155
 - Ipopt::DenseGenMatrix, 193
 - Ipopt::ExpandedMultiVectorMatrix, 231
 - Ipopt::ExpansionMatrix, 237
 - Ipopt::GenTMatrix, 268
 - Ipopt::LowRankUpdateSymMatrix, 480
 - Ipopt::Matrix, 538
 - Ipopt::MultiVectorMatrix, 557
 - Ipopt::ScaledMatrix, 728
 - Ipopt::SumMatrix, 784
 - Ipopt::SumSymMatrix, 790
 - Ipopt::SymMatrix, 797
 - Ipopt::TransposeMatrix, 861
 - Ipopt::ZeroMatrix, 914
 - Ipopt::ZeroSymMatrix, 918
- ComputeDampingIndicators
 - Ipopt::IpoptCalculatedQuantities, 365
- ComputeEigenVectors
 - Ipopt::DenseGenMatrix, 192
- ComputeFeasibilityMultipliers
 - Ipopt::IpoptAlgorithm, 336
- ComputeLUFactorInPlace
 - Ipopt::DenseGenMatrix, 192
- ComputeNewtonNormalStep
 - Ipopt::InexactNewtonNormalStep, 307
- ComputeNormalStep
 - Ipopt::InexactDoglegNormalStep, 296
 - Ipopt::InexactNormalStepCalculator, 309
- ComputeOptimalityErrorScaling
 - Ipopt::IpoptCalculatedQuantities, 365
- ComputeResidualRatio
 - Ipopt::PDFullSpaceSolver, 634
- ComputeResiduals
 - Ipopt::InexactPDSolver, 316
 - Ipopt::PDFullSpaceSolver, 634
 - Ipopt::TimingStatistics, 825
- ComputeResiduals_
 -

- Ipopt::TimingStatistics, [827](#)
- ComputeRowAMax
 - Ipopt::Matrix, [536](#)
- ComputeRowAMaxImpl
 - Ipopt::CompoundMatrix, [154](#)
 - Ipopt::CompoundSymMatrix, [164](#)
 - Ipopt::DenseGenMatrix, [193](#)
 - Ipopt::DenseSymMatrix, [198](#)
 - Ipopt::DiagMatrix, [219](#)
 - Ipopt::ExpandedMultiVectorMatrix, [230](#)
 - Ipopt::ExpansionMatrix, [236](#)
 - Ipopt::GenTMatrix, [268](#)
 - Ipopt::IdentityMatrix, [279](#)
 - Ipopt::LowRankUpdateSymMatrix, [480](#)
 - Ipopt::Matrix, [538](#)
 - Ipopt::MultiVectorMatrix, [557](#)
 - Ipopt::ScaledMatrix, [728](#)
 - Ipopt::SumMatrix, [783](#)
 - Ipopt::SumSymMatrix, [789](#)
 - Ipopt::SymScaledMatrix, [803](#)
 - Ipopt::SymTMatrix, [810](#)
 - Ipopt::TransposeMatrix, [861](#)
 - Ipopt::ZeroMatrix, [914](#)
 - Ipopt::ZeroSymMatrix, [918](#)
- ComputeSearchDirection
 - Ipopt::CGSearchDirCalculator, [149](#)
 - Ipopt::InexactSearchDirCalculator, [326](#)
 - Ipopt::IpoptAlgorithm, [336](#)
 - Ipopt::PDSearchDirCalculator, [645](#)
 - Ipopt::SearchDirectionCalculator, [733](#)
 - Ipopt::TimingStatistics, [825](#)
- ComputeSearchDirection_
 - Ipopt::TimingStatistics, [827](#)
- ComputeSymTScalingFactors
 - Ipopt::InexactTSymScalingMethod, [328](#)
 - Ipopt::Mc19TSymScalingMethod, [543](#)
 - Ipopt::SlackBasedTSymScalingMethod, [735](#)
 - Ipopt::TSymScalingMethod, [885](#)
- con_integer_md_
 - Ipopt::AmplTNLP, [87](#)
- con_numeric_md_
 - Ipopt::AmplTNLP, [87](#)
- con_string_md_
 - Ipopt::AmplTNLP, [87](#)
- config_default.h
 - COIN_HAS_AS_L, [966](#)
 - COIN_HAS_BLAS, [966](#)
 - COIN_HAS_HSL, [966](#)
 - COIN_HAS_LAPACK, [966](#)
- config_ipopt_default.h
 - IPOPT_VERSION, [967](#)
- ConsiderNewSystem
 - Ipopt::CGPerturbationHandler, [142](#)
 - Ipopt::PDPerturbationHandler, [639](#)
- consist_tol
 - ma77_control_d, [503](#)
 - ma97_control_d, [522](#)
- const_comps_
 - Ipopt::CompoundMatrix, [155](#)
 - Ipopt::CompoundSymMatrix, [164](#)
 - Ipopt::CompoundVector, [175](#)
- const_vecs_
 - Ipopt::MultiVectorMatrix, [558](#)
- ConstComp
 - Ipopt::CompoundMatrix, [155](#)
 - Ipopt::CompoundSymMatrix, [164](#)
 - Ipopt::CompoundVector, [175](#)
- ConstPtr
 - Ipopt, [50](#)
 - Ipopt::SmartPtr, [741](#)
- ConstVec
 - Ipopt::MultiVectorMatrix, [558](#)
- constr_mult_init_max_
 - Ipopt::DefaultIterateInitializer, [186](#)
 - Ipopt::RestoIterateInitializer, [715](#)
- constr_mult_reset_threshold_
 - Ipopt::MinC_1NrmRestorationPhase, [549](#)
- constr_viol_
 - Ipopt::SolveStatistics, [746](#)
- constr_viol_normtype
 - Ipopt::IpoptCalculatedQuantities, [363](#)
- constr_viol_normtype_
 - Ipopt::IpoptCalculatedQuantities, [366](#)
- constr_viol_tol_
 - Ipopt::MinC_1NrmRestorationPhase, [549](#)
 - Ipopt::OptimalityErrorConvergenceCheck, [596](#)
- Constraint_Source
 - Ipopt::AmplSuffixHandler, [76](#)
- contrib/CGPenalty/IpCGPenaltyCq.hpp, [976](#)
- contrib/CGPenalty/IpCGPenaltyData.hpp, [977](#)
- contrib/CGPenalty/IpCGPenaltyLSAcceptor.hpp, [977](#)
- contrib/CGPenalty/IpCGPenaltyRegOp.hpp, [977](#)
- contrib/CGPenalty/IpCGPerturbationHandler.hpp, [978](#)
- contrib/CGPenalty/IpCGSearchDirCalc.hpp, [978](#)
- contrib/CGPenalty/IpPiecewisePenalty.hpp, [979](#)
- contrib/LinearSolverLoader/HSLLoader.h, [979](#)
- contrib/LinearSolverLoader/LibraryHandler.h, [992](#)
- contrib/LinearSolverLoader/PardisoLoader.h, [993](#)
- control_
 - Ipopt::Ma77SolverInterface, [512](#)
 - Ipopt::Ma86SolverInterface, [520](#)
 - Ipopt::Ma97SolverInterface, [531](#)
- conv_check_
 - Ipopt::BacktrackingLineSearch, [106](#)
 - Ipopt::IpoptAlgorithm, [337](#)
- conval_called_with_current_x_
 - Ipopt::AmplTNLP, [86](#)
- ConvergenceCheck

- Ipopt::ConvergenceCheck, 181
- ConvergenceStatus
 - Ipopt::ConvergenceCheck, 181
- ConvertValues
 - Ipopt::TripletToCSRConverter, 873
- Copy
 - Ipopt::DenseGenMatrix, 191
 - Ipopt::Vector, 892
- CopyFromPos
 - Ipopt::DenseVector, 205
- CopyImpl
 - Ipopt::CompoundVector, 172
 - Ipopt::DenseVector, 205
 - Ipopt::Vector, 895
- CopyToPos
 - Ipopt::DenseVector, 205
- CopyTrialToCurrent
 - Ipopt::IpoptData, 382
- correct_bound_multiplier
 - Ipopt::IpoptAlgorithm, 336
- corrector_compl_avrg_red_fact_
 - Ipopt::FilterLSAcceptor, 254
- corrector_type_
 - Ipopt::FilterLSAcceptor, 254
- CorrectorTypeEnum
 - Ipopt::FilterLSAcceptor, 250
- count_qf_evals_
 - Ipopt::QualityFunctionMuOracle, 672
- count_restorations_
 - Ipopt::MinC_1NrmRestorationPhase, 549
- count_successive_filter_rejections_
 - Ipopt::FilterLSAcceptor, 255
- count_successive_shortened_steps_
 - Ipopt::BacktrackingLineSearch, 106
- Counter
 - Ipopt::OptionsList::OptionValue, 604
 - Ipopt::RegisteredOption, 682
- counter_
 - Ipopt::OptionsList::OptionValue, 604
 - Ipopt::RegisteredOption, 686
- counter_first_type_penalty_updates_
 - Ipopt::CGPenaltyLSAcceptor, 137
- counter_second_type_penalty_updates_
 - Ipopt::CGPenaltyLSAcceptor, 137
- cpu_time_start
 - Ipopt::IpoptData, 383
- cpu_time_start_
 - Ipopt::IpoptData, 388
- CpuTime
 - Ipopt, 52
- create_new_s
 - Ipopt::IteratesVector, 406
- create_new_s_copy
 - Ipopt::IteratesVector, 406
- create_new_v_L
 - Ipopt::IteratesVector, 409
- create_new_v_L_copy
 - Ipopt::IteratesVector, 409
- create_new_v_U
 - Ipopt::IteratesVector, 410
- create_new_v_U_copy
 - Ipopt::IteratesVector, 410
- create_new_x
 - Ipopt::IteratesVector, 405
- create_new_x_copy
 - Ipopt::IteratesVector, 405
- create_new_y_c
 - Ipopt::IteratesVector, 406
- create_new_y_c_copy
 - Ipopt::IteratesVector, 406
- create_new_y_d
 - Ipopt::IteratesVector, 407
- create_new_y_d_copy
 - Ipopt::IteratesVector, 407
- create_new_z_L
 - Ipopt::IteratesVector, 408
- create_new_z_L_copy
 - Ipopt::IteratesVector, 408
- create_new_z_U
 - Ipopt::IteratesVector, 408
- create_new_z_U_copy
 - Ipopt::IteratesVector, 408
- CreateAugmentedSpace
 - Ipopt::StdAugSystemSolver, 763
- CreateAugmentedSystem
 - Ipopt::StdAugSystemSolver, 763
- CreateBlockFromSpace
 - Ipopt::CompoundMatrix, 153
- curr
 - Ipopt::IpoptData, 380
- curr_
 - Ipopt::IpoptData, 386
- curr_Av_c_
 - Ipopt::InexactPDTerminationTester, 322
- curr_Av_d_
 - Ipopt::InexactPDTerminationTester, 322
- curr_Av_norm_
 - Ipopt::InexactPDTerminationTester, 322
- curr_DR_x_
 - Ipopt::LimMemQuasiNewtonUpdater, 455
- curr_DR_x_tag_
 - Ipopt::LimMemQuasiNewtonUpdater, 455
- curr_W_times_vec_s
 - Ipopt::InexactCq, 287
- curr_W_times_vec_s_cache_
 - Ipopt::InexactCq, 288
- curr_W_times_vec_x
 - Ipopt::InexactCq, 287

curr_W_times_vec_x_cache_
 Ipopt::InexactCq, 288
 curr_Wu_s
 Ipopt::InexactCq, 287
 curr_Wu_s_cache_
 Ipopt::InexactCq, 288
 curr_Wu_x
 Ipopt::InexactCq, 287
 curr_Wu_x_cache_
 Ipopt::InexactCq, 288
 curr_Wv_s_
 Ipopt::InexactPDTerminationTester, 323
 curr_Wv_x_
 Ipopt::InexactPDTerminationTester, 323
 curr_added_y_nrm2
 Ipopt::CGPenaltyCq, 120
 curr_added_y_nrm2_cache_
 Ipopt::CGPenaltyCq, 121
 curr_avg_compl
 Ipopt::IpoptCalculatedQuantities, 363
 curr_avg_compl_cache_
 Ipopt::IpoptCalculatedQuantities, 372
 curr_barrier_error
 Ipopt::IpoptCalculatedQuantities, 362
 curr_barrier_error_cache_
 Ipopt::IpoptCalculatedQuantities, 372
 curr_barrier_obj
 Ipopt::IpoptCalculatedQuantities, 357
 curr_barrier_obj_cache_
 Ipopt::IpoptCalculatedQuantities, 367
 curr_c
 Ipopt::IpoptCalculatedQuantities, 357
 curr_c_amax_
 Ipopt::QualityFunctionMuOracle, 673
 curr_c_asum_
 Ipopt::QualityFunctionMuOracle, 673
 curr_c_cache_
 Ipopt::IpoptCalculatedQuantities, 368
 curr_c_nrm2_
 Ipopt::QualityFunctionMuOracle, 673
 curr_centality_measure
 Ipopt::IpoptCalculatedQuantities, 362
 curr_centality_measure_cache_
 Ipopt::IpoptCalculatedQuantities, 372
 curr_cg_pert_fact
 Ipopt::CGPenaltyCq, 120
 curr_cg_pert_fact_cache_
 Ipopt::CGPenaltyCq, 121
 curr_compl_s_L
 Ipopt::IpoptCalculatedQuantities, 360
 curr_compl_s_L_cache_
 Ipopt::IpoptCalculatedQuantities, 370
 curr_compl_s_U
 Ipopt::IpoptCalculatedQuantities, 360
 curr_compl_s_U_cache_
 Ipopt::IpoptCalculatedQuantities, 370
 curr_compl_x_L
 Ipopt::IpoptCalculatedQuantities, 360
 curr_compl_x_L_cache_
 Ipopt::IpoptCalculatedQuantities, 370
 curr_compl_x_U
 Ipopt::IpoptCalculatedQuantities, 360
 curr_compl_x_U_cache_
 Ipopt::IpoptCalculatedQuantities, 370
 curr_complementarity
 Ipopt::IpoptCalculatedQuantities, 361
 curr_complementarity_cache_
 Ipopt::IpoptCalculatedQuantities, 372
 curr_constraint_violation
 Ipopt::IpoptCalculatedQuantities, 359
 curr_constraint_violation_cache_
 Ipopt::IpoptCalculatedQuantities, 369
 curr_d
 Ipopt::IpoptCalculatedQuantities, 358
 curr_d_cache_
 Ipopt::IpoptCalculatedQuantities, 368
 curr_d_minus_s
 Ipopt::IpoptCalculatedQuantities, 358
 curr_d_minus_s_amax_
 Ipopt::QualityFunctionMuOracle, 673
 curr_d_minus_s_asum_
 Ipopt::QualityFunctionMuOracle, 673
 curr_d_minus_s_cache_
 Ipopt::IpoptCalculatedQuantities, 368
 curr_d_minus_s_nrm2_
 Ipopt::QualityFunctionMuOracle, 673
 curr_direct_deriv_penalty_function
 Ipopt::CGPenaltyCq, 120
 curr_direct_deriv_penalty_function_cache_
 Ipopt::CGPenaltyCq, 121
 curr_dual_frac_to_the_bound
 Ipopt::IpoptCalculatedQuantities, 363
 curr_dual_infeasibility
 Ipopt::IpoptCalculatedQuantities, 361
 curr_dual_infeasibility_cache_
 Ipopt::IpoptCalculatedQuantities, 371
 curr_eta_
 Ipopt::CGPenaltyLSAceptor, 137
 Ipopt::LimMemQuasiNewtonUpdater, 455
 curr_exact_hessian
 Ipopt::IpoptCalculatedQuantities, 359
 curr_exact_hessian_cache_
 Ipopt::IpoptCalculatedQuantities, 369
 curr_f
 Ipopt::IpoptCalculatedQuantities, 356
 curr_f_cache_
 Ipopt::IpoptCalculatedQuantities, 367
 curr_fast_direct_deriv_penalty_function

Ipopt::CGPenaltyCq, [120](#)
 curr_fast_direct_deriv_penalty_function_cache_
 Ipopt::CGPenaltyCq, [121](#)
 curr_grad_barrier_obj_s
 Ipopt::IpoptCalculatedQuantities, [357](#)
 curr_grad_barrier_obj_s_
 Ipopt::InexactPDTerminationTester, [322](#)
 curr_grad_barrier_obj_s_cache_
 Ipopt::IpoptCalculatedQuantities, [367](#)
 curr_grad_barrier_obj_x
 Ipopt::IpoptCalculatedQuantities, [357](#)
 curr_grad_barrier_obj_x_
 Ipopt::InexactPDTerminationTester, [322](#)
 curr_grad_barrier_obj_x_cache_
 Ipopt::IpoptCalculatedQuantities, [367](#)
 curr_grad_f
 Ipopt::IpoptCalculatedQuantities, [357](#)
 curr_grad_f_cache_
 Ipopt::IpoptCalculatedQuantities, [367](#)
 curr_grad_lag_s
 Ipopt::IpoptCalculatedQuantities, [360](#)
 curr_grad_lag_s_amax_
 Ipopt::QualityFunctionMuOracle, [673](#)
 curr_grad_lag_s_asum_
 Ipopt::QualityFunctionMuOracle, [673](#)
 curr_grad_lag_s_cache_
 Ipopt::IpoptCalculatedQuantities, [370](#)
 curr_grad_lag_s_nrm2_
 Ipopt::QualityFunctionMuOracle, [673](#)
 curr_grad_lag_with_damping_s
 Ipopt::IpoptCalculatedQuantities, [360](#)
 curr_grad_lag_with_damping_s_cache_
 Ipopt::IpoptCalculatedQuantities, [370](#)
 curr_grad_lag_with_damping_x
 Ipopt::IpoptCalculatedQuantities, [360](#)
 curr_grad_lag_with_damping_x_cache_
 Ipopt::IpoptCalculatedQuantities, [370](#)
 curr_grad_lag_x
 Ipopt::IpoptCalculatedQuantities, [360](#)
 curr_grad_lag_x_amax_
 Ipopt::QualityFunctionMuOracle, [673](#)
 curr_grad_lag_x_asum_
 Ipopt::QualityFunctionMuOracle, [672](#)
 curr_grad_lag_x_cache_
 Ipopt::IpoptCalculatedQuantities, [370](#)
 curr_grad_lag_x_nrm2_
 Ipopt::QualityFunctionMuOracle, [673](#)
 curr_gradBarrTDelta
 Ipopt::IpoptCalculatedQuantities, [363](#)
 curr_gradBarrTDelta_cache_
 Ipopt::IpoptCalculatedQuantities, [373](#)
 curr_jac_c
 Ipopt::IpoptCalculatedQuantities, [358](#)
 curr_jac_c_
 Ipopt::InexactPDTerminationTester, [322](#)
 curr_jac_c_cache_
 Ipopt::IpoptCalculatedQuantities, [368](#)
 curr_jac_c_times_vec
 Ipopt::IpoptCalculatedQuantities, [359](#)
 curr_jac_c_times_vec_cache_
 Ipopt::IpoptCalculatedQuantities, [369](#)
 curr_jac_cT_times_curr_y_c
 Ipopt::IpoptCalculatedQuantities, [358](#)
 curr_jac_cT_times_vec
 Ipopt::IpoptCalculatedQuantities, [358](#)
 curr_jac_cT_times_vec_cache_
 Ipopt::IpoptCalculatedQuantities, [369](#)
 curr_jac_cd_norm
 Ipopt::CGPenaltyCq, [120](#)
 curr_jac_cd_norm_cache_
 Ipopt::CGPenaltyCq, [121](#)
 curr_jac_cdT_times_curr_cdminuss
 Ipopt::InexactCq, [286](#)
 curr_jac_cdT_times_curr_cdminuss_cache_
 Ipopt::InexactCq, [288](#)
 curr_jac_d
 Ipopt::IpoptCalculatedQuantities, [358](#)
 curr_jac_d_
 Ipopt::InexactPDTerminationTester, [322](#)
 curr_jac_d_cache_
 Ipopt::IpoptCalculatedQuantities, [368](#)
 curr_jac_d_times_vec
 Ipopt::IpoptCalculatedQuantities, [359](#)
 curr_jac_d_times_vec_cache_
 Ipopt::IpoptCalculatedQuantities, [369](#)
 curr_jac_dT_times_curr_y_d
 Ipopt::IpoptCalculatedQuantities, [359](#)
 curr_jac_dT_times_vec
 Ipopt::IpoptCalculatedQuantities, [358](#)
 curr_jac_dT_times_vec_cache_
 Ipopt::IpoptCalculatedQuantities, [369](#)
 curr_jac_times_normal_c
 Ipopt::InexactCq, [287](#)
 curr_jac_times_normal_c_cache_
 Ipopt::InexactCq, [289](#)
 curr_jac_times_normal_d
 Ipopt::InexactCq, [287](#)
 curr_jac_times_normal_d_cache_
 Ipopt::InexactCq, [289](#)
 curr_kkt_penalty
 Ipopt::CGPenaltyData, [127](#)
 curr_kkt_penalty_
 Ipopt::CGPenaltyData, [128](#)
 curr_lm_memory_
 Ipopt::LimMemQuasiNewtonUpdater, [456](#)
 curr_lm_memory_old_
 Ipopt::LimMemQuasiNewtonUpdater, [457](#)
 curr_mu

Ipopt::IpoptData, 382
 curr_mu_
 Ipopt::IpoptData, 386
 curr_nabla_phi_plus_ATy_s_
 Ipopt::InexactPDTerminationTester, 322
 curr_nabla_phi_plus_ATy_x_
 Ipopt::InexactPDTerminationTester, 322
 curr_nlp_constraint_violation
 Ipopt::IpoptCalculatedQuantities, 359
 curr_nlp_constraint_violation_cache_
 Ipopt::IpoptCalculatedQuantities, 369
 curr_nlp_error
 Ipopt::IpoptCalculatedQuantities, 362
 curr_nlp_error_cache_
 Ipopt::IpoptCalculatedQuantities, 372
 curr_nu
 Ipopt::InexactData, 292
 curr_nu_
 Ipopt::InexactData, 293
 curr_obj_val_
 Ipopt::OptimalityErrorConvergenceCheck, 597
 curr_omega_
 Ipopt::InexactDoglegNormalStep, 296
 curr_penalty
 Ipopt::CGPenaltyData, 126
 curr_penalty_
 Ipopt::CGPenaltyData, 128
 curr_penalty_function
 Ipopt::CGPenaltyCq, 120
 curr_penalty_function_cache_
 Ipopt::CGPenaltyCq, 121
 curr_penalty_pert_
 Ipopt::CGPenaltyData, 128
 curr_primal_dual_system_error
 Ipopt::IpoptCalculatedQuantities, 362
 curr_primal_dual_system_error_cache_
 Ipopt::IpoptCalculatedQuantities, 372
 curr_primal_frac_to_the_bound
 Ipopt::IpoptCalculatedQuantities, 362
 curr_primal_infeasibility
 Ipopt::IpoptCalculatedQuantities, 361
 curr_primal_infeasibility_cache_
 Ipopt::IpoptCalculatedQuantities, 371
 curr_red_DR_x_
 Ipopt::LimMemQuasiNewtonUpdater, 455
 curr_relaxed_compl_s_L
 Ipopt::IpoptCalculatedQuantities, 361
 curr_relaxed_compl_s_L_cache_
 Ipopt::IpoptCalculatedQuantities, 371
 curr_relaxed_compl_s_U
 Ipopt::IpoptCalculatedQuantities, 361
 curr_relaxed_compl_s_U_cache_
 Ipopt::IpoptCalculatedQuantities, 371
 curr_relaxed_compl_x_L
 Ipopt::IpoptCalculatedQuantities, 361
 curr_relaxed_compl_x_L_cache_
 Ipopt::IpoptCalculatedQuantities, 371
 curr_relaxed_compl_x_U
 Ipopt::IpoptCalculatedQuantities, 361
 curr_relaxed_compl_x_U_cache_
 Ipopt::IpoptCalculatedQuantities, 371
 curr_scaled_A_norm2
 Ipopt::InexactCq, 287
 curr_scaled_Ac_norm
 Ipopt::InexactCq, 287
 curr_scaled_Ac_norm_cache_
 Ipopt::InexactCq, 288
 curr_scaled_y_Amax
 Ipopt::CGPenaltyCq, 120
 curr_scaled_y_Amax_cache_
 Ipopt::CGPenaltyCq, 121
 curr_scaling_slacks
 Ipopt::InexactCq, 287
 curr_scaling_slacks_
 Ipopt::InexactPDTerminationTester, 322
 curr_scaling_slacks_cache_
 Ipopt::InexactCq, 288
 curr_sigma_s
 Ipopt::IpoptCalculatedQuantities, 363
 curr_sigma_s_cache_
 Ipopt::IpoptCalculatedQuantities, 372
 curr_sigma_x
 Ipopt::IpoptCalculatedQuantities, 363
 curr_sigma_x_cache_
 Ipopt::IpoptCalculatedQuantities, 372
 curr_slack_s_L
 Ipopt::IpoptCalculatedQuantities, 356
 curr_slack_s_L_
 Ipopt::QualityFunctionMuOracle, 672
 curr_slack_s_L_cache_
 Ipopt::IpoptCalculatedQuantities, 366
 curr_slack_s_U
 Ipopt::IpoptCalculatedQuantities, 356
 curr_slack_s_U_
 Ipopt::QualityFunctionMuOracle, 672
 curr_slack_s_U_cache_
 Ipopt::IpoptCalculatedQuantities, 366
 curr_slack_scaled_d_minus_s
 Ipopt::InexactCq, 287
 curr_slack_scaled_d_minus_s_cache_
 Ipopt::InexactCq, 288
 curr_slack_x_L
 Ipopt::IpoptCalculatedQuantities, 356
 curr_slack_x_L_
 Ipopt::QualityFunctionMuOracle, 672
 curr_slack_x_L_cache_
 Ipopt::IpoptCalculatedQuantities, 366
 curr_slack_x_U

- Ipopt::IpoptCalculatedQuantities, 356
 - curr_slack_x_U_
 - Ipopt::QualityFunctionMuOracle, 672
 - curr_slack_x_U_cache_
 - Ipopt::IpoptCalculatedQuantities, 366
 - curr_tau
 - Ipopt::IpoptData, 382
 - curr_tau_
 - Ipopt::IpoptData, 387
 - curr_tt1_norm_
 - Ipopt::InexactPDTerminationTester, 322
 - curr_tt2_norm_
 - Ipopt::InexactPDTerminationTester, 323
 - curr_uWu
 - Ipopt::InexactCq, 287
 - curr_uWu_cache_
 - Ipopt::InexactCq, 289
 - curr_v_L_
 - Ipopt::QualityFunctionMuOracle, 672
 - curr_v_U_
 - Ipopt::QualityFunctionMuOracle, 672
 - curr_z_L_
 - Ipopt::QualityFunctionMuOracle, 672
 - curr_z_U_
 - Ipopt::QualityFunctionMuOracle, 672
 - CurrPenaltyPert
 - Ipopt::CGPenaltyData, 126
 - current_level_
 - Ipopt::Ma97SolverInterface, 532
 - current_registering_category_
 - Ipopt::RegisteredOptions, 693
 - CurrentIsAcceptable
 - Ipopt::BacktrackingLineSearch, 103
 - Ipopt::ConvergenceCheck, 182
 - Ipopt::OptimalityErrorConvergenceCheck, 595
 - CurrentIsBest
 - Ipopt::CGPenaltyLSAcceptor, 135
 - CurrentPerturbation
 - Ipopt::CGPerturbationHandler, 143
 - Ipopt::PDPerturbationHandler, 640
 - custom_solver_
 - Ipopt::AlgorithmBuilder, 65
 - Ipopt::InexactAlgorithmBuilder, 283
- D
- d
 - Ipopt::IpoptNLP, 395
 - Ipopt::OrigIpoptNLP, 611
 - Ipopt::RestIpoptNLP, 705
- DEGENERATE
 - Ipopt::CGPerturbationHandler, 142
 - Ipopt::PDPerturbationHandler, 638
- DIVERGING
 - Ipopt::ConvergenceCheck, 181
- DIVERGING_ITERATES
 - Ipopt, 48
- DUAL_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, 100
- DUAL_AND_FULL_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, 100
- D_
 - Ipopt::LimMemQuasiNewtonUpdater, 456
 - Ipopt::LowRankUpdateSymMatrix, 480
- d_L
 - Ipopt::IpoptNLP, 396
 - Ipopt::OrigIpoptNLP, 612
 - Ipopt::RestIpoptNLP, 706
- d_L_
 - Ipopt::OrigIpoptNLP, 618
 - Ipopt::RestIpoptNLP, 710
- d_U
 - Ipopt::IpoptNLP, 396
 - Ipopt::OrigIpoptNLP, 613
 - Ipopt::RestIpoptNLP, 706
- d_U_
 - Ipopt::OrigIpoptNLP, 618
 - Ipopt::RestIpoptNLP, 711
- D_c_ext_
 - Ipopt::LowRankSSAugSystemSolver, 476
- d_c_tag_
 - Ipopt::GenAugSystemSolver, 260
 - Ipopt::LowRankAugSystemSolver, 468
 - Ipopt::LowRankSSAugSystemSolver, 475
 - Ipopt::StdAugSystemSolver, 765
- d_cache_
 - Ipopt::OrigIpoptNLP, 617
- d_d_tag_
 - Ipopt::GenAugSystemSolver, 261
 - Ipopt::LowRankAugSystemSolver, 469
 - Ipopt::LowRankSSAugSystemSolver, 475
 - Ipopt::StdAugSystemSolver, 765
- d_eval_time
 - Ipopt::OrigIpoptNLP, 615
- d_eval_time_
 - Ipopt::OrigIpoptNLP, 620
- d_evals
 - Ipopt::IpoptNLP, 397
 - Ipopt::OrigIpoptNLP, 614
 - Ipopt::RestIpoptNLP, 708
- d_evals_
 - Ipopt::OrigIpoptNLP, 620
 - Ipopt::RestIpoptNLP, 712
- d_l_space_
 - Ipopt::OrigIpoptNLP, 616
 - Ipopt::RestIpoptNLP, 710
 - Ipopt::TNLPAdapter, 848
- D_old_
 - Ipopt::LimMemQuasiNewtonUpdater, 458

- d_s_tag_
 - Ipopt::GenAugSystemSolver, [260](#)
 - Ipopt::LowRankAugSystemSolver, [468](#)
 - Ipopt::LowRankSSAugSystemSolver, [474](#)
 - Ipopt::StdAugSystemSolver, [765](#)
- d_space_
 - Ipopt::OrigIpoptNLP, [616](#)
 - Ipopt::RestIpoptNLP, [709](#)
 - Ipopt::TNLPAdapter, [848](#)
- d_space_orig_
 - Ipopt::NLPBoundsRemover, [580](#)
- d_u_space_
 - Ipopt::OrigIpoptNLP, [616](#)
 - Ipopt::RestIpoptNLP, [710](#)
 - Ipopt::TNLPAdapter, [849](#)
- D_x_plus_wr_d
 - Ipopt::AugRestoSystemSolver, [91](#)
- d_x_plus_wr_d_cache_
 - Ipopt::AugRestoSystemSolver, [92](#)
- d_x_tag_
 - Ipopt::GenAugSystemSolver, [260](#)
 - Ipopt::LowRankAugSystemSolver, [468](#)
 - Ipopt::LowRankSSAugSystemSolver, [474](#)
 - Ipopt::StdAugSystemSolver, [765](#)
- DBG_ASSERT
 - IpDebug.hpp, [968](#)
- DBG_DO
 - IpDebug.hpp, [968](#)
- DBG_EXEC
 - IpDebug.hpp, [968](#)
- DBG_PRINT
 - IpDebug.hpp, [968](#)
- DBG_PRINT_MATRIX
 - IpDebug.hpp, [968](#)
 - IpMatrix.hpp, [1015](#)
- DBG_PRINT_VECTOR
 - IpDebug.hpp, [968](#)
 - IpVector.hpp, [1018](#)
- DBG_START_FUN
 - IpDebug.hpp, [968](#)
- DBG_START_METH
 - IpDebug.hpp, [968](#)
- DBG_VERBOSE
 - IpDebug.hpp, [968](#)
- DPARAM_
 - Ipopt::IterativePardisoSolverInterface, [426](#)
 - Ipopt::IterativeWsmSolverInterface, [436](#)
 - Ipopt::PardisoSolverInterface, [630](#)
 - Ipopt::WsmSolverInterface, [911](#)
- DR_x
 - Ipopt::RestIpoptNLP, [708](#)
- DR_x_
 - Ipopt::RestIpoptNLP, [711](#)
- DRS_
 - Ipopt::LimMemQuasiNewtonUpdater, [457](#)
- DRS_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [459](#)
- dT_times_barH_times_d
 - Ipopt::CGPenaltyCq, [120](#)
- dampind_s_L_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- dampind_s_U_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- dampind_x_L_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- dampind_x_U_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- dc_vals_copy_
 - Ipopt::GenAugSystemSolver, [261](#)
- dd_vals_copy_
 - Ipopt::GenAugSystemSolver, [261](#)
- debug_cnt_
 - Ipopt::IterativePardisoSolverInterface, [427](#)
 - Ipopt::PardisoSolverInterface, [630](#)
- debug_last_iter_
 - Ipopt::IterativePardisoSolverInterface, [426](#)
 - Ipopt::PardisoSolverInterface, [630](#)
- DebugPrint
 - Ipopt::DependentResult, [215](#)
- DebugPrintCachedResults
 - Ipopt::CachedResults, [116](#)
- decomposition_type_
 - Ipopt::InexactSearchDirCalculator, [327](#)
- DecompositionTypeEnum
 - Ipopt::InexactSearchDirCalculator, [325](#)
- decr_factor_
 - Ipopt::IterativePardisoSolverInterface, [425](#)
- default_number_
 - Ipopt::RegisteredOption, [686](#)
- default_string_
 - Ipopt::RegisteredOption, [686](#)
- DefaultInteger
 - Ipopt::RegisteredOption, [684](#)
- DefaultIterateInitializer
 - Ipopt::DefaultIterateInitializer, [185](#)
- DefaultNumber
 - Ipopt::RegisteredOption, [684](#)
- DefaultString
 - Ipopt::RegisteredOption, [684](#)
- DefaultStringAsEnum
 - Ipopt::RegisteredOption, [684](#)
- degen_iters_
 - Ipopt::CGPerturbationHandler, [145](#)
 - Ipopt::PDPerturbationHandler, [641](#)
- degen_iters_max_
 - Ipopt::CGPerturbationHandler, [146](#)
 - Ipopt::PDPerturbationHandler, [642](#)
- DegenType

- Ipopt::CGPerturbationHandler, 142
- Ipopt::PDPerturbationHandler, 638
- DeleteAllJournals
 - Ipopt::Journalist, 443
- delta
 - Ipopt::IpoptData, 380
- delta_
 - Ipopt::FilterLSAcceptor, 253
 - Ipopt::IpoptData, 386
- delta_aff
 - Ipopt::IpoptData, 381
- delta_aff_
 - Ipopt::IpoptData, 386
- delta_c_
 - Ipopt::GenAugSystemSolver, 261
 - Ipopt::LowRankAugSystemSolver, 468
 - Ipopt::LowRankSSAugSystemSolver, 475
 - Ipopt::StdAugSystemSolver, 765
- delta_c_curr_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 641
- delta_c_last_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 640
- delta_cd
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 640
- delta_cd_exp_
 - Ipopt::CGPerturbationHandler, 146
 - Ipopt::PDPerturbationHandler, 642
- delta_cd_val_
 - Ipopt::CGPerturbationHandler, 146
 - Ipopt::PDPerturbationHandler, 642
- delta_cgfast
 - Ipopt::CGPenaltyData, 125
- delta_cgfast_
 - Ipopt::CGPenaltyData, 127
- delta_cgpen
 - Ipopt::CGPenaltyData, 125
- delta_cgpen_
 - Ipopt::CGPenaltyData, 127
- delta_d_
 - Ipopt::GenAugSystemSolver, 261
 - Ipopt::LowRankAugSystemSolver, 469
 - Ipopt::LowRankSSAugSystemSolver, 475
 - Ipopt::StdAugSystemSolver, 766
- delta_d_curr_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 641
- delta_d_last_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 640
- delta_s_
 - Ipopt::GenAugSystemSolver, 260
 - Ipopt::LowRankAugSystemSolver, 468
 - Ipopt::LowRankSSAugSystemSolver, 474
 - Ipopt::StdAugSystemSolver, 765
- delta_s_curr_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 641
- delta_s_last_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 640
- delta_x_
 - Ipopt::GenAugSystemSolver, 260
 - Ipopt::LowRankAugSystemSolver, 468
 - Ipopt::LowRankSSAugSystemSolver, 474
 - Ipopt::StdAugSystemSolver, 765
- delta_x_curr_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 640
- delta_x_last_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 640
- delta_xs_dec_fact_
 - Ipopt::CGPerturbationHandler, 145
 - Ipopt::PDPerturbationHandler, 642
- delta_xs_first_inc_fact_
 - Ipopt::CGPerturbationHandler, 145
 - Ipopt::PDPerturbationHandler, 642
- delta_xs_inc_fact_
 - Ipopt::CGPerturbationHandler, 145
 - Ipopt::PDPerturbationHandler, 642
- delta_xs_init_
 - Ipopt::CGPerturbationHandler, 145
 - Ipopt::PDPerturbationHandler, 642
- delta_xs_max_
 - Ipopt::CGPerturbationHandler, 145
 - Ipopt::PDPerturbationHandler, 641
- delta_xs_min_
 - Ipopt::CGPerturbationHandler, 145
 - Ipopt::PDPerturbationHandler, 641
- delta_y_max_
 - Ipopt::CGSearchDirCalculator, 150
- DenseGenMatrix
 - Ipopt::DenseGenMatrix, 190, 191
- DenseGenMatrixSpace
 - Ipopt::DenseGenMatrixSpace, 195
- DenseSymMatrix
 - Ipopt::DenseSymMatrix, 197
- DenseSymMatrixSpace
 - Ipopt::DenseSymMatrixSpace, 200
- DenseVector
 - Ipopt::DenseVector, 204
- DenseVectorSpace
 - Ipopt::DenseVectorSpace, 211
- dependency_detection_with_rhs_
 - Ipopt::TNLPAdapter, 847

- dependency_detector_
 - Ipopt::TNLPAdapter, 845
- dependent_tags_
 - Ipopt::DependentResult, 216
- DependentResult
 - Ipopt::DependentResult, 214, 215
- DependentsIdentical
 - Ipopt::DependentResult, 215
- derivative_test_
 - Ipopt::TNLPAdapter, 846
- derivative_test_first_index_
 - Ipopt::TNLPAdapter, 846
- derivative_test_perturbation_
 - Ipopt::TNLPAdapter, 846
- derivative_test_print_all_
 - Ipopt::TNLPAdapter, 846
- derivative_test_tol_
 - Ipopt::TNLPAdapter, 846
- DerivativeTestEnum
 - Ipopt::TNLPAdapter, 841
- Description
 - Ipopt::AmplOptionsList::AmplOption, 71
- description_
 - Ipopt::AmplOptionsList::AmplOption, 71
 - Ipopt::RegisteredOption::string_entry, 778
- DetachObserver
 - Ipopt::Subject, 780
- DetectTinyStep
 - Ipopt::BacktrackingLineSearch, 103
- DetermineDependentConstraints
 - Ipopt::TNLPAdapter, 845
- DetermineDependentRows
 - Ipopt::Ma28TDDependencyDetector, 493
 - Ipopt::Ma77SolverInterface, 511
 - Ipopt::Ma86SolverInterface, 520
 - Ipopt::Ma97SolverInterface, 530
 - Ipopt::MumpsSolverInterface, 564
 - Ipopt::SparseSymLinearSolverInterface, 752
 - Ipopt::TDependencyDetector, 818
 - Ipopt::TSymDependencyDetector, 876
 - Ipopt::TSymLinearSolver, 881
 - Ipopt::WsmpSolverInterface, 909
- DetermineScaling
 - Ipopt::NLPScalingObject, 587
 - Ipopt::StandardScalingBase, 758
- DetermineScalingParametersImpl
 - Ipopt::EquilibrationScaling, 225
 - Ipopt::GradientScaling, 274
 - Ipopt::NoNLPScalingObject, 589
 - Ipopt::StandardScalingBase, 758
 - Ipopt::UserScaling, 886
- detlog
 - ma77_info_d, 504
 - ma86_info_d, 514
- detsign
 - ma77_info_d, 504
 - ma86_info_d, 514
- df_
 - Ipopt::StandardScalingBase, 759
- diag_
 - Ipopt::DiagMatrix, 219
- diag_space_c_
 - Ipopt::StdAugSystemSolver, 764
- diag_space_d_
 - Ipopt::StdAugSystemSolver, 764
- diag_space_s_
 - Ipopt::StdAugSystemSolver, 764
- diag_space_x_
 - Ipopt::StdAugSystemSolver, 764
- DiagMatrix
 - Ipopt::DiagMatrix, 218
- DiagMatrixSpace
 - Ipopt::DiagMatrixSpace, 220, 221
- diagnostics_level
 - ma86_control_d, 512
- Diagonal
 - Ipopt::CompoundMatrixSpace, 159
- diagonal_
 - Ipopt::CompoundMatrixSpace, 160
- Dim
 - Ipopt::IdentityMatrix, 279
 - Ipopt::SymMatrix, 797
 - Ipopt::SymMatrixSpace, 799
 - Ipopt::Vector, 894
 - Ipopt::VectorSpace, 901
- dim_
 - Ipopt::Filter, 244
 - Ipopt::IterativePardisoSolverInterface, 423
 - Ipopt::IterativeWsmpSolverInterface, 435
 - Ipopt::Ma27TSolverInterface, 489
 - Ipopt::Ma57TSolverInterface, 498
 - Ipopt::PardisoSolverInterface, 628
 - Ipopt::PiecewisePenalty, 657
 - Ipopt::TripletToCSRConverter, 874
 - Ipopt::TSymLinearSolver, 881
 - Ipopt::VectorSpace, 901
 - Ipopt::WsmpSolverInterface, 909
- dimensions_set_
 - Ipopt::CompoundMatrixSpace, 159
 - Ipopt::CompoundSymMatrixSpace, 168
- DimensionsSet
 - Ipopt::CompoundMatrixSpace, 159
 - Ipopt::CompoundSymMatrixSpace, 168
- Diverging_iterates
 - IpReturnCodes_inc.h, 997
- diverging_iterates_tol_
 - Ipopt, 48
- diverging_iterates_tol_
 - Ipopt::OptimalityErrorConvergenceCheck, 597

- DoBacktrackingLineSearch
 - Ipopt::BacktrackingLineSearch, [101](#)
- DoFallback
 - Ipopt::BacktrackingLSAcceptor, [111](#)
 - Ipopt::CGPenaltyLSAcceptor, [134](#)
- Dominated
 - Ipopt::FilterEntry, [246](#)
- dont_print_
 - Ipopt::OptionsList::OptionValue, [605](#)
- DontPrint
 - Ipopt::OptionsList::OptionValue, [604](#)
- Dot
 - Ipopt::Vector, [892](#)
- dot_cache_
 - Ipopt::Vector, [898](#)
- DotImpl
 - Ipopt::CompoundVector, [173](#)
 - Ipopt::DenseVector, [205](#)
 - Ipopt::Vector, [895](#)
- dr_x_
 - Ipopt::RestoIpoptNLP, [711](#)
- ds_vals_copy_
 - Ipopt::GenAugSystemSolver, [261](#)
- dual_frac_to_the_bound
 - Ipopt::IpoptCalculatedQuantities, [362](#)
- dual_frac_to_the_bound_cache_
 - Ipopt::IpoptCalculatedQuantities, [372](#)
- dual_inf_
 - Ipopt::SolveStatistics, [746](#)
- dual_inf_tol_
 - Ipopt::OptimalityErrorConvergenceCheck, [596](#)
- dummy_cache_
 - Ipopt::PDFullSpaceSolver, [634](#)
- dump_
 - Ipopt::Ma97SolverInterface, [532](#)
- duplicate
 - mc68_info, [545](#)
- dx_
 - Ipopt::StandardScalingBase, [759](#)
- dx_vals_copy_
 - Ipopt::GenAugSystemSolver, [261](#)
- E
- ERROR_IN_STEP_COMPUTATION
 - Ipopt, [48](#)
- EXACT
 - Ipopt, [45](#)
- EJournalCategory
 - Ipopt, [46](#)
- EJournalLevel
 - Ipopt, [46](#)
- EMatrixFormat
 - Ipopt::SparseSymLinearSolverInterface, [750](#)
- ENormType
 - Ipopt, [45](#)
- ESymSolverStatus
 - Ipopt, [46](#)
- ETerminationTest
 - Ipopt::IterativeSolverTerminationTester, [429](#)
- ETriFull
 - Ipopt::TripletToCSRConverter, [872](#)
- ElementWiseAbs
 - Ipopt::Vector, [893](#)
- ElementWiseAbsImpl
 - Ipopt::CompoundVector, [174](#)
 - Ipopt::DenseVector, [206](#)
 - Ipopt::Vector, [896](#)
- ElementWiseDivide
 - Ipopt::Vector, [893](#)
- ElementWiseDivideImpl
 - Ipopt::CompoundVector, [173](#)
 - Ipopt::DenseVector, [206](#)
 - Ipopt::Vector, [896](#)
- ElementWiseMax
 - Ipopt::Vector, [893](#)
- ElementWiseMaxImpl
 - Ipopt::CompoundVector, [173](#)
 - Ipopt::DenseVector, [206](#)
 - Ipopt::Vector, [896](#)
- ElementWiseMin
 - Ipopt::Vector, [893](#)
- ElementWiseMinImpl
 - Ipopt::CompoundVector, [173](#)
 - Ipopt::DenseVector, [206](#)
 - Ipopt::Vector, [896](#)
- ElementWiseMultiply
 - Ipopt::Vector, [893](#)
- ElementWiseMultiplyImpl
 - Ipopt::CompoundVector, [173](#)
 - Ipopt::DenseVector, [206](#)
 - Ipopt::Vector, [896](#)
- ElementWiseReciprocal
 - Ipopt::Vector, [893](#)
- ElementWiseReciprocalImpl
 - Ipopt::CompoundVector, [174](#)
 - Ipopt::DenseVector, [206](#)
 - Ipopt::Vector, [896](#)
- ElementWiseSgn
 - Ipopt::Vector, [893](#)
- ElementWiseSgnImpl
 - Ipopt::CompoundVector, [174](#)
 - Ipopt::DenseVector, [207](#)
 - Ipopt::Vector, [896](#)
- ElementWiseSqrt
 - Ipopt::Vector, [893](#)
- ElementWiseSqrtImpl
 - Ipopt::CompoundVector, [174](#)
 - Ipopt::DenseVector, [207](#)

- Ipopt::Vector, [896](#)
- End
 - Ipopt::TimedTask, [821](#)
- end_called_
 - Ipopt::TimedTask, [822](#)
- EndIfStarted
 - Ipopt::TimedTask, [821](#)
- epsilon_c_
 - Ipopt::CGPenaltyLSAcceptor, [136](#)
- eq_mult_calculator_
 - Ipopt::DefaultIterateInitializer, [187](#)
 - Ipopt::MinC_1NrmRestorationPhase, [549](#)
- eq_multiplier_calculator_
 - Ipopt::IpoptAlgorithm, [337](#)
- EqMultiplierCalculator
 - Ipopt::EqMultiplierCalculator, [222](#)
- EquilibrationScaling
 - Ipopt::EquilibrationScaling, [224](#), [225](#)
- Error_In_Step_Computation
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- Eta
 - Ipopt::RestIpoptNLP, [708](#)
- eta_
 - Ipopt::InexactLSAcceptor, [303](#)
 - Ipopt::PenaltyLSAcceptor, [653](#)
- eta_factor_
 - Ipopt::RestIpoptNLP, [711](#)
- eta_min_
 - Ipopt::CGPenaltyLSAcceptor, [135](#)
- eta_mu_exponent_
 - Ipopt::RestIpoptNLP, [711](#)
- eta_penalty_
 - Ipopt::CGPenaltyLSAcceptor, [135](#)
- eta_phi_
 - Ipopt::FilterLSAcceptor, [253](#)
- Eval_F_CB
 - IpStdCInterface.h, [1002](#)
- Eval_G_CB
 - IpStdCInterface.h, [1002](#)
- Eval_Grad_F_CB
 - IpStdCInterface.h, [1002](#)
- Eval_H_CB
 - IpStdCInterface.h, [1002](#)
- Eval_Jac_G_CB
 - IpStdCInterface.h, [1002](#)
- Eval_c
 - Ipopt::NLP, [573](#)
 - Ipopt::NLPBoundsRemover, [579](#)
 - Ipopt::TNLPAdapter, [843](#)
- Eval_d
 - Ipopt::NLP, [573](#)
 - Ipopt::NLPBoundsRemover, [579](#)
 - Ipopt::TNLPAdapter, [843](#)
- Eval_f
 - Ipopt::NLP, [573](#)
 - Ipopt::NLPBoundsRemover, [578](#)
 - Ipopt::TNLPAdapter, [843](#)
- eval_f
 - Ipopt::AmplTNLP, [82](#)
 - Ipopt::StdInterfaceTNLP, [771](#)
 - Ipopt::TNLP, [834](#)
 - Ipopt::TNLPReducer, [856](#)
 - IpStdCInterface.h, [1005](#)
- eval_f_
 - Ipopt::StdInterfaceTNLP, [773](#)
- eval_g
 - Ipopt::AmplTNLP, [83](#)
 - Ipopt::StdInterfaceTNLP, [771](#)
 - Ipopt::TNLP, [834](#)
 - Ipopt::TNLPReducer, [856](#)
 - IpStdCInterface.h, [1005](#)
- eval_g_
 - Ipopt::StdInterfaceTNLP, [773](#)
- Eval_grad_f
 - Ipopt::NLP, [573](#)
 - Ipopt::NLPBoundsRemover, [578](#)
 - Ipopt::TNLPAdapter, [843](#)
- eval_grad_f
 - Ipopt::AmplTNLP, [82](#)
 - Ipopt::StdInterfaceTNLP, [771](#)
 - Ipopt::TNLP, [834](#)
 - Ipopt::TNLPReducer, [856](#)
 - IpStdCInterface.h, [1005](#)
- eval_grad_f_
 - Ipopt::StdInterfaceTNLP, [773](#)
- Eval_h
 - Ipopt::NLP, [574](#)
 - Ipopt::NLPBoundsRemover, [579](#)
 - Ipopt::TNLPAdapter, [843](#)
- eval_h
 - Ipopt::AmplTNLP, [83](#)
 - Ipopt::StdInterfaceTNLP, [771](#)
 - Ipopt::TNLP, [834](#)
 - Ipopt::TNLPReducer, [856](#)
 - IpStdCInterface.h, [1005](#)
- eval_h_
 - Ipopt::StdInterfaceTNLP, [774](#)
- Eval_jac_c
 - Ipopt::NLP, [573](#)
 - Ipopt::NLPBoundsRemover, [579](#)
 - Ipopt::TNLPAdapter, [843](#)
- Eval_jac_d
 - Ipopt::NLP, [573](#)
 - Ipopt::NLPBoundsRemover, [579](#)
 - Ipopt::TNLPAdapter, [843](#)
- eval_jac_g
 - Ipopt::AmplTNLP, [83](#)

- Ipopt::StdInterfaceTNLP, 771
 - Ipopt::TNLP, 834
 - Ipopt::TNLPReducer, 856
 - IpStdCInterface.h, 1005
- eval_jac_g_
 - Ipopt::StdInterfaceTNLP, 774
- evaluate_orig_obj_at_resto_trial_
 - Ipopt::RestIpoptNLP, 711
- ExactHessianUpdater
 - Ipopt::ExactHessianUpdater, 227
- exp_matrix_
 - Ipopt::ExpandedMultiVectorMatrixSpace, 233
- expanded_pos_
 - Ipopt::ExpansionMatrixSpace, 239
- expanded_values_
 - Ipopt::DenseVector, 208
- expanded_vu_
 - Ipopt::LowRankSSAugSystemSolver, 476
- ExpandedMultiVectorMatrix
 - Ipopt::ExpandedMultiVectorMatrix, 229
- ExpandedMultiVectorMatrixOwnerSpace
 - Ipopt::ExpandedMultiVectorMatrix, 230
- ExpandedMultiVectorMatrixSpace
 - Ipopt::ExpandedMultiVectorMatrixSpace, 232
- ExpandedPosIndices
 - Ipopt::ExpansionMatrix, 236
 - Ipopt::ExpansionMatrixSpace, 239
- ExpandedValues
 - Ipopt::DenseVector, 205
- ExpansionMatrix
 - Ipopt::ExpansionMatrix, 235
- ExpansionMatrixSpace
 - Ipopt::ExpansionMatrixSpace, 238
- expect_infeasible_problem_
 - Ipopt::BacktrackingLineSearch, 104
 - Ipopt::MinC_1NrmRestorationPhase, 549
- expect_infeasible_problem_ctol_
 - Ipopt::BacktrackingLineSearch, 104
- expect_infeasible_problem_ytol_
 - Ipopt::BacktrackingLineSearch, 104
- F
- f
 - Ipopt::IpoptNLP, 395, 397
 - Ipopt::OrigIpoptNLP, 611
 - Ipopt::RestIpoptNLP, 705
- FAILED
 - Ipopt::ConvergenceCheck, 181
- FEASIBLE_POINT_FOUND
 - Ipopt, 48
- FILTER_OBJ_CONSTR
 - Ipopt::AdaptiveMuUpdate, 58
- FIRST_ORDER_TEST
 - Ipopt::TNLPAdapter, 842
- FORTTRAN_STYLE
 - Ipopt::TNLP, 832
- FULL_STEP_FOR_Y
 - Ipopt::BacktrackingLineSearch, 100
- f_array_in
 - mc68_control, 544
- f_array_out
 - mc68_control, 544
- f_arrays
 - ma77_control_d, 501
 - ma86_control_d, 512
 - ma97_control_d, 521
- f_cache_
 - Ipopt::OrigIpoptNLP, 617
- f_eval_time
 - Ipopt::OrigIpoptNLP, 615
- f_eval_time_
 - Ipopt::OrigIpoptNLP, 620
- f_evals
 - Ipopt::IpoptNLP, 397
 - Ipopt::OrigIpoptNLP, 613
 - Ipopt::RestIpoptNLP, 708
- f_evals_
 - Ipopt::OrigIpoptNLP, 620
 - Ipopt::RestIpoptNLP, 712
- FALSE
 - IpStdCInterface.h, 1001
- factor_
 - Ipopt::IdentityMatrix, 279
- factor_min
 - ma97_control_d, 522
- Factorization
 - Ipopt::DenseGenMatrix, 190
 - Ipopt::IterativePardisoSolverInterface, 423
 - Ipopt::IterativeWsmpSolverInterface, 434
 - Ipopt::Ma27TSolverInterface, 488
 - Ipopt::Ma57TSolverInterface, 498
 - Ipopt::MumpsSolverInterface, 564
 - Ipopt::PardisoSolverInterface, 628
 - Ipopt::WsmpSolverInterface, 909
- factorization_
 - Ipopt::DenseGenMatrix, 193
- factorizations_since_recomputed_ordering_
 - Ipopt::WsmpSolverInterface, 911
- factors_
 - Ipopt::SumMatrix, 784
 - Ipopt::SumSymMatrix, 790
- fallback_activated_
 - Ipopt::BacktrackingLineSearch, 106
- fast_des_fact_
 - Ipopt::CGSearchDirCalculator, 150
- fast_step_computation_
 - Ipopt::PDSearchDirCalculator, 645
- ftidx_

- Ipopt::Ma97SolverInterface, [531](#)
- Feasible_Point_Found
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [48](#)
- file_
 - Ipopt::FileJournal, [242](#)
- file_name
 - IpStdCInterface.h, [1005](#)
- file_name_
 - Ipopt::IpoptException, [391](#)
- file_size
 - ma77_control_d, [502](#)
- FileJournal
 - Ipopt::FileJournal, [241](#)
- FillIdentity
 - Ipopt::DenseGenMatrix, [191](#)
 - Ipopt::DenseSymMatrix, [198](#)
- FillRowCol
 - Ipopt::TripletHelper, [868](#)
- FillRowCol_
 - Ipopt::TripletHelper, [868–870](#)
- FillStruct
 - Ipopt::SymTMatrix, [809](#)
- FillValues
 - Ipopt::SymTMatrix, [810](#)
 - Ipopt::TripletHelper, [868](#)
- FillValues_
 - Ipopt::TripletHelper, [869, 870](#)
- FillValuesFromVector
 - Ipopt::TripletHelper, [868](#)
- FillWithNewVectors
 - Ipopt::MultiVectorMatrix, [556](#)
- Filter
 - Ipopt::Filter, [243](#)
- filter_
 - Ipopt::AdaptiveMuUpdate, [62](#)
 - Ipopt::FilterLSAcceptor, [255](#)
- filter_list_
 - Ipopt::Filter, [244](#)
- filter_margin_fact_
 - Ipopt::AdaptiveMuUpdate, [61](#)
- filter_max_margin_
 - Ipopt::AdaptiveMuUpdate, [61](#)
- filter_reset_trigger_
 - Ipopt::FilterLSAcceptor, [255](#)
- FilterEntry
 - Ipopt::FilterEntry, [245, 246](#)
- FilterLSAcceptor
 - Ipopt::FilterLSAcceptor, [250](#)
- FinalObjective
 - Ipopt::SolveStatistics, [745](#)
- FinalScaledObjective
 - Ipopt::SolveStatistics, [745](#)
- finalize_metadata
 - Ipopt::TNLP, [835](#)
- finalize_solution
 - Ipopt::AmplTNLP, [83](#)
 - Ipopt::StdInterfaceTNLP, [772](#)
 - Ipopt::TNLP, [835](#)
 - Ipopt::TNLPReducer, [856](#)
- finalize_test
 - Ipopt::CGPerturbationHandler, [144](#)
 - Ipopt::PDPerturbationHandler, [640](#)
- FinalizeSolution
 - Ipopt::IpoptNLP, [398](#)
 - Ipopt::NLP, [574](#)
 - Ipopt::NLPBoundsRemover, [579](#)
 - Ipopt::OrigIpoptNLP, [614](#)
 - Ipopt::RestoIpoptNLP, [704](#)
 - Ipopt::TNLPAdapter, [844](#)
- find_tag
 - Ipopt::OptionsList, [601](#)
- FindAcceptableTrialPoint
 - Ipopt::BacktrackingLineSearch, [100](#)
 - Ipopt::LineSearch, [460](#)
- findiff_jac_ia_
 - Ipopt::TNLPAdapter, [851](#)
- findiff_jac_ja_
 - Ipopt::TNLPAdapter, [851](#)
- findiff_jac_nnz_
 - Ipopt::TNLPAdapter, [851](#)
- findiff_jac_postriple_
 - Ipopt::TNLPAdapter, [851](#)
- findiff_perturbation_
 - Ipopt::TNLPAdapter, [847](#)
- findiff_x_l_
 - Ipopt::TNLPAdapter, [851](#)
- findiff_x_u_
 - Ipopt::TNLPAdapter, [851](#)
- first_call_
 - Ipopt::LowRankAugSystemSolver, [469](#)
 - Ipopt::LowRankSSAugSystemSolver, [475](#)
- first_iter_resto_
 - Ipopt::MonotoneMuUpdate, [553](#)
- first_resto_iter_
 - Ipopt::RestoConvergenceCheck, [696](#)
- fix_mu_oracle_
 - Ipopt::AdaptiveMuUpdate, [62](#)
- fixed_variable_treatment_
 - Ipopt::TNLPAdapter, [846](#)
- FixedVariableTreatmentEnum
 - Ipopt::TNLPAdapter, [841](#)
- fkeep_
 - Ipopt::Ma97SolverInterface, [531](#)
- flag
 - ma77_info_d, [504](#)
 - ma86_info_d, [514](#)
 - ma97_info, [523](#)

- mc68_info, 545
- flag68
 - ma97_info, 523
- flag77
 - ma97_info, 524
- flexible_penalty_function_
 - Ipopt::InexactLSAcceptor, 303
- FlushBuffer
 - Ipopt::Journal, 439
 - Ipopt::Journalist, 443
- FlushBufferImpl
 - Ipopt::FileJournal, 241
 - Ipopt::Journal, 439
 - Ipopt::StreamJournal, 777
- FracToBound
 - Ipopt::Vector, 894
- FracToBoundImpl
 - Ipopt::CompoundVector, 175
 - Ipopt::DenseVector, 207
 - Ipopt::Vector, 897
- free_mu_mode_
 - Ipopt::IpoptData, 387
- free_mu_oracle_
 - Ipopt::AdaptiveMuUpdate, 62
- FreeInternalStorage
 - Ipopt::DenseVectorSpace, 211
 - Ipopt::GenTMatrixSpace, 271
 - Ipopt::SymTMatrixSpace, 813
- FreeMuMode
 - Ipopt::IpoptData, 383
- Full_Format
 - Ipopt::TripletToCSRConverter, 872
- full_g_
 - Ipopt::TNLPAdapter, 849
- full_lambda_
 - Ipopt::TNLPAdapter, 849
- full_step_accepted
 - Ipopt::InexactData, 292
- full_step_accepted_
 - Ipopt::InexactData, 293
- full_x_
 - Ipopt::TNLPAdapter, 849
- G
- g
 - IpStdCInterface.h, 1006
- g_L
 - IpStdCInterface.h, 1004
- g_L_
 - Ipopt::StdInterfaceTNLP, 772
- g_U
 - IpStdCInterface.h, 1004
- g_U_
 - Ipopt::StdInterfaceTNLP, 772
- g_keep_map_
 - Ipopt::TNLPReducer, 857
- g_scaling
 - IpStdCInterface.h, 1006
- g_scaling_
 - Ipopt::StdInterfaceTNLP, 774
- g_sol_
 - Ipopt::AmplTNLP, 86
 - Ipopt::StdInterfaceTNLP, 775
- gamma_hat_
 - Ipopt::CGPenaltyLSAcceptor, 135
- gamma_phi_
 - Ipopt::FilterLSAcceptor, 253
- gamma_theta_
 - Ipopt::FilterLSAcceptor, 254
- gamma_tilde_
 - Ipopt::CGPenaltyLSAcceptor, 136
- GenAugSystemSolver
 - Ipopt::GenAugSystemSolver, 258
- GenKKTsSolverInterface
 - Ipopt::GenKKTsSolverInterface, 262
- GenTMatrix
 - Ipopt::GenTMatrix, 266, 267
 - Ipopt::GenTMatrixSpace, 271
- GenTMatrixSpace
 - Ipopt::GenTMatrixSpace, 270
- get_bounds_info
 - Ipopt::AmplTNLP, 82
 - Ipopt::StdInterfaceTNLP, 770
 - Ipopt::TNLP, 833
 - Ipopt::TNLPReducer, 855
- get_constraints_linearity
 - Ipopt::AmplTNLP, 82
 - Ipopt::TNLP, 833
 - Ipopt::TNLPReducer, 855
- get_deltas_for_wrong_inertia
 - Ipopt::CGPerturbationHandler, 143
 - Ipopt::PDPerturbationHandler, 640
- get_deltas_for_wrong_inertia_called_
 - Ipopt::CGPerturbationHandler, 144
 - Ipopt::PDPerturbationHandler, 641
- get_discrete_info
 - Ipopt::AmplTNLP, 84
- get_list_of_nonlinear_variables
 - Ipopt::AmplTNLP, 83
 - Ipopt::TNLP, 835
 - Ipopt::TNLPReducer, 857
- get_nlp_info
 - Ipopt::AmplTNLP, 82
 - Ipopt::StdInterfaceTNLP, 770
 - Ipopt::TNLP, 832
 - Ipopt::TNLPReducer, 855
- get_number_of_nonlinear_variables
 - Ipopt::AmplTNLP, 83

- [Ipopt::TNLP](#), 835
 - [Ipopt::TNLPReducer](#), 857
- [get_options](#)
 - [Ipopt::AmplTNLP](#), 85
- [get_scaling_parameters](#)
 - [Ipopt::AmplTNLP](#), 83
 - [Ipopt::StdInterfaceTNLP](#), 770
 - [Ipopt::TNLP](#), 833
 - [Ipopt::TNLPReducer](#), 855
- [get_starting_point](#)
 - [Ipopt::AmplTNLP](#), 82
 - [Ipopt::StdInterfaceTNLP](#), 770
 - [Ipopt::TNLP](#), 833
 - [Ipopt::TNLPReducer](#), 855
- [get_suffix_handler](#)
 - [Ipopt::AmplTNLP](#), 84
- [get_unscaled_x](#)
 - [Ipopt::OrigIpoptNLP](#), 615
- [get_var_con_metadata](#)
 - [Ipopt::AmplTNLP](#), 82
 - [Ipopt::TNLP](#), 832
- [get_variables_linearity](#)
 - [Ipopt::TNLP](#), 833
 - [Ipopt::TNLPReducer](#), 855
- [get_warm_start_iterate](#)
 - [Ipopt::TNLP](#), 834
 - [Ipopt::TNLPReducer](#), 855
- [GetBlockCols](#)
 - [Ipopt::CompoundMatrixSpace](#), 158
- [GetBlockDim](#)
 - [Ipopt::CompoundSymMatrixSpace](#), 167
- [GetBlockRows](#)
 - [Ipopt::CompoundMatrixSpace](#), 158
- [GetBoolValue](#)
 - [Ipopt::OptionsList](#), 601
- [GetBoundsInformation](#)
 - [Ipopt::NLP](#), 573
 - [Ipopt::NLPBoundsRemover](#), 578
 - [Ipopt::TNLPAdapter](#), 843
- [GetCachedResult](#)
 - [Ipopt::CachedResults](#), 114, 115
- [GetCachedResult1Dep](#)
 - [Ipopt::CachedResults](#), 115
- [GetCachedResult2Dep](#)
 - [Ipopt::CachedResults](#), 115
- [GetCachedResult3Dep](#)
 - [Ipopt::CachedResults](#), 115, 116
- [GetComp](#)
 - [Ipopt::CompoundMatrix](#), 153
 - [Ipopt::CompoundSymMatrix](#), 163
 - [Ipopt::CompoundVector](#), 172
- [GetCompNonConst](#)
 - [Ipopt::CompoundMatrix](#), 153
 - [Ipopt::CompoundSymMatrix](#), 163
- [Ipopt::CompoundVector](#), 172
- [GetCompSpace](#)
 - [Ipopt::CompoundMatrixSpace](#), 158
 - [Ipopt::CompoundSymMatrixSpace](#), 167
 - [Ipopt::CompoundVectorSpace](#), 179
- [GetDiag](#)
 - [Ipopt::DiagMatrix](#), 218
 - [Ipopt::LowRankUpdateSymMatrix](#), 479
- [GetEnumValue](#)
 - [Ipopt::OptionsList](#), 601
- [GetExpansionMatrix](#)
 - [Ipopt::ExpandedMultiVectorMatrix](#), 230
 - [Ipopt::ExpandedMultiVectorMatrixSpace](#), 233
- [GetFactor](#)
 - [Ipopt::IdentityMatrix](#), 278
- [GetIntegerMetaData](#)
 - [Ipopt::DenseVectorSpace](#), 212
- [GetIntegerSuffixValues](#)
 - [Ipopt::AmplSuffixHandler](#), 76
- [GetIntegerValue](#)
 - [Ipopt::OptionsList](#), 601
- [GetIpoptNLP](#)
 - [Ipopt::IpoptCalculatedQuantities](#), 363
- [GetIterateFromComp](#)
 - [Ipopt::IteratesVector](#), 411
- [GetJnlst](#)
 - [Ipopt::IterativeSolverTerminationTester](#), 430
- [GetJournal](#)
 - [Ipopt::Journalist](#), 443
- [GetNonConstIterateFromComp](#)
 - [Ipopt::IteratesVector](#), 411
- [GetNumberEntries](#)
 - [Ipopt::TripletHelper](#), 868
- [GetNumberEntries_](#)
 - [Ipopt::TripletHelper](#), 868
- [GetNumberSuffixValues](#)
 - [Ipopt::AmplSuffixHandler](#), 76
- [GetNumericMetaData](#)
 - [Ipopt::DenseVectorSpace](#), 212
- [GetNumericValue](#)
 - [Ipopt::OptionsList](#), 601
- [GetOption](#)
 - [Ipopt::RegisteredOptions](#), 692
- [getPDPert](#)
 - [Ipopt::IpoptData](#), 385
- [GetQuasiNewtonApproximationSpaces](#)
 - [Ipopt::NLP](#), 574
 - [Ipopt::NLPBoundsRemover](#), 580
 - [Ipopt::TNLPAdapter](#), 844
- [GetRawPtr](#)
 - [Ipopt](#), 50
 - [Ipopt::SmartPtr](#), 741
- [GetResult](#)
 - [Ipopt::DependentResult](#), 215

- GetScalingParameters
 - Ipopt::NLP, [574](#)
 - Ipopt::NLPBoundsRemover, [579](#)
 - Ipopt::TNLPAdapter, [844](#)
- GetSolverIterations
 - Ipopt::InexactNormalTerminationTester, [312](#)
 - Ipopt::InexactPDTerminationTester, [320](#)
 - Ipopt::IterativeSolverTerminationTester, [430](#)
- GetSpaces
 - Ipopt::IpoptNLP, [396](#)
 - Ipopt::NLP, [572](#)
 - Ipopt::NLPBoundsRemover, [578](#)
 - Ipopt::OrigIpoptNLP, [613](#)
 - Ipopt::RestIpoptNLP, [707](#)
 - Ipopt::TNLPAdapter, [842](#)
- GetStartingPoint
 - Ipopt::NLP, [573](#)
 - Ipopt::NLPBoundsRemover, [578](#)
 - Ipopt::TNLPAdapter, [843](#)
- GetStringMetaData
 - Ipopt::DenseVectorSpace, [211](#), [212](#)
- GetStringValue
 - Ipopt::OptionsList, [601](#)
- GetTag
 - Ipopt::TaggedObject, [816](#)
- GetTagSum
 - Ipopt::IteratesVector, [411](#)
- GetTerm
 - Ipopt::SumMatrix, [783](#)
 - Ipopt::SumSymMatrix, [789](#)
- GetTermSpace
 - Ipopt::SumMatrixSpace, [786](#)
 - Ipopt::SumSymMatrixSpace, [792](#)
- GetU
 - Ipopt::LowRankUpdateSymMatrix, [479](#)
- GetUnscaledMatrix
 - Ipopt::ScaledMatrix, [727](#)
 - Ipopt::SymScaledMatrix, [802](#)
- GetUnscaledMatrixNonConst
 - Ipopt::ScaledMatrix, [727](#)
 - Ipopt::SymScaledMatrix, [802](#)
- GetV
 - Ipopt::LowRankUpdateSymMatrix, [479](#)
- GetValidStrings
 - Ipopt::RegisteredOption, [684](#)
- GetValue
 - Ipopt::OptionsList::OptionValue, [604](#)
- GetValuesArrayPtr
 - Ipopt::IterativePardisoSolverInterface, [422](#)
 - Ipopt::IterativeWsmvSolverInterface, [433](#)
 - Ipopt::Ma27TSolverInterface, [487](#)
 - Ipopt::Ma57TSolverInterface, [497](#)
 - Ipopt::Ma77SolverInterface, [510](#)
 - Ipopt::Ma86SolverInterface, [518](#)
 - Ipopt::Ma97SolverInterface, [529](#)
 - Ipopt::MumpsSolverInterface, [563](#)
 - Ipopt::PardisoSolverInterface, [627](#)
 - Ipopt::SparseSymLinearSolverInterface, [751](#)
 - Ipopt::WsmvSolverInterface, [908](#)
- GetVector
 - Ipopt::ExpandedMultiVectorMatrix, [230](#)
 - Ipopt::MultiVectorMatrix, [556](#)
- GetVectorNonConst
 - Ipopt::MultiVectorMatrix, [556](#)
- GetVectors
 - Ipopt::IterativeSolverTerminationTester, [430](#)
- GetWarmStartIterate
 - Ipopt::IpoptNLP, [395](#)
 - Ipopt::NLP, [573](#)
 - Ipopt::NLPBoundsRemover, [578](#)
 - Ipopt::OrigIpoptNLP, [611](#)
 - Ipopt::RestIpoptNLP, [704](#)
 - Ipopt::TNLPAdapter, [843](#)
- GiveMatrixToSolver
 - Ipopt::TSymLinearSolver, [881](#)
- grad_f
 - Ipopt::IpoptNLP, [395](#), [398](#)
 - Ipopt::OrigIpoptNLP, [611](#)
 - Ipopt::RestIpoptNLP, [705](#)
- grad_f_cache_
 - Ipopt::OrigIpoptNLP, [617](#)
- grad_f_eval_time
 - Ipopt::OrigIpoptNLP, [615](#)
- grad_f_eval_time_
 - Ipopt::OrigIpoptNLP, [620](#)
- grad_f_evals
 - Ipopt::IpoptNLP, [397](#)
 - Ipopt::OrigIpoptNLP, [614](#)
 - Ipopt::RestIpoptNLP, [708](#)
- grad_f_evals_
 - Ipopt::OrigIpoptNLP, [620](#)
 - Ipopt::RestIpoptNLP, [712](#)
- grad_kappa_times_damping_s
 - Ipopt::IpoptCalculatedQuantities, [357](#)
- grad_kappa_times_damping_s_cache_
 - Ipopt::IpoptCalculatedQuantities, [368](#)
- grad_kappa_times_damping_x
 - Ipopt::IpoptCalculatedQuantities, [357](#)
- grad_kappa_times_damping_x_cache_
 - Ipopt::IpoptCalculatedQuantities, [368](#)
- GradientScaling
 - Ipopt::GradientScaling, [273](#)
- H
- h
 - Ipopt::IpoptNLP, [395](#), [398](#)
 - Ipopt::OrigIpoptNLP, [612](#)
 - Ipopt::RestIpoptNLP, [705](#)

h_cache_
 lpopt::OriglpoptNLP, 617
 h_eval_time
 lpopt::OriglpoptNLP, 615
 h_eval_time_
 lpopt::OriglpoptNLP, 621
 h_evals
 lpopt::lpoptNLP, 397
 lpopt::OriglpoptNLP, 614
 lpopt::RestlpoptNLP, 708
 h_evals_
 lpopt::OriglpoptNLP, 620
 lpopt::RestlpoptNLP, 712
 h_idx_map_
 lpopt::TNLPAdapter, 851
 h_space_
 lpopt::LimMemQuasiNewtonUpdater, 454
 lpopt::OriglpoptNLP, 617
 lpopt::RestlpoptNLP, 710
 HSLLoader.h
 ipfint, 986
 LSL_HSLLibraryName, 991
 LSL_isHSLLoaded, 990
 LSL_isMA27available, 990
 LSL_isMA28available, 990
 LSL_isMA57available, 990
 LSL_isMA77available, 990
 LSL_isMA86available, 991
 LSL_isMA97available, 991
 LSL_isMC19available, 991
 LSL_isMC68available, 991
 LSL_loadHSL, 989
 LSL_setMA27, 991
 LSL_setMA28, 991
 LSL_setMA57, 991
 LSL_setMA77, 991
 LSL_setMA86, 992
 LSL_setMA97, 992
 LSL_setMC19, 992
 LSL_setMC68, 992
 LSL_unloadHSL, 990
 ma27ad_t, 986
 ma27bd_t, 986
 ma27cd_t, 986
 ma27id_t, 986
 ma28ad_t, 986
 ma57ad_t, 986
 ma57bd_t, 986
 ma57cd_t, 986
 ma57ed_t, 986
 ma57id_t, 986
 ma77_alter, 984
 ma77_alter_t, 988
 ma77_analyse, 983
 ma77_analyse_t, 987
 ma77_control, 983
 ma77_default_control, 983
 ma77_default_control_t, 986
 ma77_enquire_indef, 984
 ma77_enquire_indef_t, 988
 ma77_enquire_posdef, 984
 ma77_enquire_posdef_t, 987
 ma77_factor, 983
 ma77_factor_solve, 983
 ma77_factor_solve_t, 987
 ma77_factor_t, 987
 ma77_finalise, 984
 ma77_finalise_t, 988
 ma77_info, 983
 ma77_input_reals, 983
 ma77_input_reals_t, 987
 ma77_input_vars, 983
 ma77_input_vars_t, 987
 ma77_open, 983
 ma77_open_nelt, 983
 ma77_open_nelt_t, 987
 ma77_open_t, 987
 ma77_resid, 983
 ma77_resid_t, 987
 ma77_restart, 984
 ma77_restart_t, 988
 ma77_scale, 984
 ma77_scale_t, 987
 ma77_solve, 983
 ma77_solve_t, 987
 ma77pkgtype_d_, 985
 ma86_analyse, 984
 ma86_analyse_t, 988
 ma86_control, 984
 ma86_default_control, 984
 ma86_default_control_t, 988
 ma86_factor, 984
 ma86_factor_solve, 984
 ma86_factor_solve_t, 988
 ma86_factor_t, 988
 ma86_finalise, 984
 ma86_finalise_t, 988
 ma86_info, 984
 ma86_solve, 984
 ma86_solve_t, 988
 ma86pkgtype_d_, 985
 ma86realtype_d_, 985
 ma97_analyse, 985
 ma97_analyse_t, 988
 ma97_control, 985
 ma97_default_control, 985
 ma97_default_control_t, 988
 ma97_factor, 985

- ma97_factor_solve, [985](#)
- ma97_factor_solve_t, [989](#)
- ma97_factor_t, [989](#)
- ma97_finalise, [985](#)
- ma97_finalise_t, [989](#)
- ma97_free_akeep, [985](#)
- ma97_free_akeep_t, [989](#)
- ma97_info, [985](#)
- ma97_solve, [985](#)
- ma97_solve_t, [989](#)
- ma97pkgtype_d_, [985](#)
- ma97realtype_d_, [985](#)
- mc19ad_t, [989](#)
- mc68_default_control_t, [989](#)
- mc68_order_t, [989](#)
- HaltOnError_Option
 - Ipopt::AmplOptionsList, [73](#)
- has_lower_
 - Ipopt::RegisteredOption, [686](#)
- has_upper_
 - Ipopt::RegisteredOption, [686](#)
- HasChanged
 - Ipopt::TaggedObject, [816](#)
- HasComputeAlphaForY
 - Ipopt::BacktrackingLSAcceptor, [111](#)
 - Ipopt::InexactLSAcceptor, [302](#)
- HasIntegerMetaData
 - Ipopt::DenseVectorSpace, [211](#)
- HasLower
 - Ipopt::RegisteredOption, [682](#)
- HasNumericMetaData
 - Ipopt::DenseVectorSpace, [211](#)
- HasStringMetaData
 - Ipopt::DenseVectorSpace, [211](#)
- HasUpper
 - Ipopt::RegisteredOption, [683](#)
- HasValidNumbers
 - Ipopt::Matrix, [536](#)
 - Ipopt::Vector, [894](#)
- HasValidNumbersImpl
 - Ipopt::CompoundMatrix, [154](#)
 - Ipopt::CompoundSymMatrix, [164](#)
 - Ipopt::CompoundVector, [175](#)
 - Ipopt::DenseGenMatrix, [193](#)
 - Ipopt::DenseSymMatrix, [198](#)
 - Ipopt::DiagMatrix, [219](#)
 - Ipopt::ExpandedMultiVectorMatrix, [230](#)
 - Ipopt::GenTMatrix, [268](#)
 - Ipopt::IdentityMatrix, [279](#)
 - Ipopt::LowRankUpdateSymMatrix, [480](#)
 - Ipopt::Matrix, [538](#)
 - Ipopt::MultiVectorMatrix, [557](#)
 - Ipopt::ScaledMatrix, [727](#)
 - Ipopt::SumMatrix, [783](#)
 - Ipopt::SumSymMatrix, [789](#)
 - Ipopt::SymScaledMatrix, [803](#)
 - Ipopt::SymTMatrix, [810](#)
 - Ipopt::TransposeMatrix, [861](#)
 - Ipopt::Vector, [897](#)
- have_affine_deltas_
 - Ipopt::IpoptData, [386](#)
- have_c_scaling
 - Ipopt::NLPScalingObject, [587](#)
 - Ipopt::StandardScalingBase, [758](#)
- have_cgfast_deltas_
 - Ipopt::CGPenaltyData, [127](#)
- have_cgpen_deltas_
 - Ipopt::CGPenaltyData, [127](#)
- have_d_scaling
 - Ipopt::NLPScalingObject, [587](#)
 - Ipopt::StandardScalingBase, [758](#)
- have_deltas_
 - Ipopt::IpoptData, [386](#)
- have_prototypes_
 - Ipopt::IpoptData, [387](#)
- have_structure_
 - Ipopt::TSymLinearSolver, [882](#)
- have_symbolic_factorization_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
 - Ipopt::IterativeWsmvSolverInterface, [436](#)
 - Ipopt::MumpsSolverInterface, [566](#)
 - Ipopt::PardisoSolverInterface, [629](#)
 - Ipopt::WsmvSolverInterface, [911](#)
- have_x_scaling
 - Ipopt::NLPScalingObject, [587](#)
 - Ipopt::StandardScalingBase, [758](#)
- HaveAddCq
 - Ipopt::IpoptCalculatedQuantities, [355](#)
- HaveAddData
 - Ipopt::IpoptData, [385](#)
- HaveAffineDeltas
 - Ipopt::IpoptData, [381](#)
- HaveCgFastDeltas
 - Ipopt::CGPenaltyData, [125](#)
- HaveCgPenDeltas
 - Ipopt::CGPenaltyData, [125](#)
- HaveDeltas
 - Ipopt::IpoptData, [381](#)
- HaveIpData
 - Ipopt::AlgorithmStrategyObject, [69](#)
- hess_degenerate_
 - Ipopt::CGPerturbationHandler, [145](#)
 - Ipopt::PDPerturbationHandler, [641](#)
- Hess_lagrangian_space_
 - Ipopt::TNLPAdapter, [849](#)
- hesset_called_
 - Ipopt::AmplTNLP, [86](#)
- hessian_approximation_

- Ipopt::OrigIpoptNLP, [619](#)
 - Ipopt::RestIpoptNLP, [712](#)
 - Ipopt::TNLPAdapter, [846](#)
- hessian_approximation_space_
 - Ipopt::OrigIpoptNLP, [619](#)
- hessian_constant_
 - Ipopt::OrigIpoptNLP, [619](#)
- hessian_updater_
 - Ipopt::IpoptAlgorithm, [337](#)
- HessianApproximationSpace
 - Ipopt, [45](#)
- HessianApproximationType
 - Ipopt, [45](#)
- HessianMatrixSpace
 - Ipopt::IpoptNLP, [396](#)
 - Ipopt::OrigIpoptNLP, [613](#)
 - Ipopt::RestIpoptNLP, [707](#)
- HessianRequiresChange
 - Ipopt::InexactPDSolver, [316](#)
- HessianUpdater
 - Ipopt::HessianUpdater, [276](#)
- hf_
 - Ipopt::TripletToCSRConverter, [873](#)
- HighRankUpdate
 - Ipopt::DenseSymMatrix, [198](#)
- HighRankUpdateTranspose
 - Ipopt::DenseGenMatrix, [191](#)
 - Ipopt::DenseSymMatrix, [198](#)
- homogeneous_
 - Ipopt::DenseVector, [209](#)
- honor_original_bounds_
 - Ipopt::OrigIpoptNLP, [619](#)
- hsl_ma77d.h
 - ma77_alter, [950](#)
 - ma77_alter_d, [951](#)
 - ma77_analyse, [949](#), [951](#)
 - ma77_control, [949](#)
 - ma77_default_control, [949](#)
 - ma77_default_control_d, [950](#)
 - ma77_enquire_indef, [950](#)
 - ma77_enquire_indef_d, [951](#)
 - ma77_enquire_posdef, [950](#)
 - ma77_enquire_posdef_d, [951](#)
 - ma77_factor, [949](#)
 - ma77_factor_d, [951](#)
 - ma77_factor_solve, [949](#)
 - ma77_factor_solve_d, [951](#)
 - ma77_finalise, [950](#)
 - ma77_finalise_d, [951](#)
 - ma77_info, [949](#)
 - ma77_input_reals, [949](#)
 - ma77_input_reals_d, [951](#)
 - ma77_input_vars, [949](#), [950](#)
 - ma77_multiply, [950](#)
 - ma77_multiply_d, [951](#)
 - ma77_open, [949](#)
 - ma77_open_d, [950](#)
 - ma77_open_nelt, [949](#), [950](#)
 - ma77_resid, [950](#)
 - ma77_resid_d, [951](#)
 - ma77_restart, [950](#)
 - ma77_restart_d, [951](#)
 - ma77_scale, [950](#)
 - ma77_scale_d, [951](#)
 - ma77_solve, [949](#)
 - ma77_solve_d, [951](#)
 - ma77_solve_fredholm, [950](#)
 - ma77_solve_fredholm_d, [951](#)
 - ma77pkgtype_d, [950](#)
- hsl_ma86d.h
 - ma86_analyse, [952](#)
 - ma86_analyse_d, [953](#)
 - ma86_control, [952](#)
 - ma86_default_control, [952](#)
 - ma86_default_control_d, [953](#)
 - ma86_factor, [952](#)
 - ma86_factor_d, [953](#)
 - ma86_factor_solve, [952](#)
 - ma86_factor_solve_d, [953](#)
 - ma86_finalise, [953](#)
 - ma86_finalise_d, [953](#)
 - ma86_info, [952](#)
 - ma86_solve, [952](#)
 - ma86_solve_d, [953](#)
 - ma86pkgtype_d, [953](#)
 - ma86realtype_d, [953](#)
- hsl_ma97d.h
 - ma97_alter, [955](#)
 - ma97_alter_d, [956](#)
 - ma97_analyse, [955](#)
 - ma97_analyse_coord, [955](#)
 - ma97_analyse_coord_d, [956](#)
 - ma97_analyse_d, [956](#)
 - ma97_control, [954](#)
 - ma97_default_control, [955](#)
 - ma97_default_control_d, [956](#)
 - ma97_enquire_indef, [955](#)
 - ma97_enquire_indef_d, [956](#)
 - ma97_enquire_posdef, [955](#)
 - ma97_enquire_posdef_d, [956](#)
 - ma97_factor, [955](#)
 - ma97_factor_d, [956](#)
 - ma97_factor_solve, [955](#)
 - ma97_factor_solve_d, [956](#)
 - ma97_finalise, [955](#)
 - ma97_finalise_d, [956](#)
 - ma97_free_akeep, [955](#)
 - ma97_free_akeep_d, [956](#)

- ma97_free_fkeep, 955
- ma97_free_fkeep_d, 956
- ma97_info, 954
- ma97_lmultiply, 955
- ma97_lmultiply_d, 957
- ma97_solve, 955
- ma97_solve_d, 956
- ma97_solve_fredholm, 955
- ma97_solve_fredholm_d, 956
- ma97_sparse_fwd_solve, 956
- ma97_sparse_fwd_solve_d, 957
- ma97pkgtype_d, 956
- ma97realtype_d, 956
- hsl_mc68i.h
 - mc68_control, 957
 - mc68_default_control, 957, 958
 - mc68_info, 957
 - mc68_order, 957, 958
- I
- INTERNAL
 - Ipopt::IterationOutput, 416
- INTERNAL_ERROR
 - Ipopt, 48
- INVALID_NUMBER_DETECTED
 - Ipopt, 48
- INVALID_OPTION
 - Ipopt, 48
- i_pos_triplet_
 - Ipopt::TripletToCSRConverter::TripletEntry, 866
- i_row_
 - Ipopt::TripletToCSRConverter::TripletEntry, 866
- IA
 - Ipopt::TripletToCSRConverter, 873
- INVP_
 - Ipopt::WsmpSolverInterface, 912
- IPARM_
 - Ipopt::IterativePardisoSolverInterface, 426
 - Ipopt::IterativeWsmpSolverInterface, 436
 - Ipopt::PardisoSolverInterface, 630
 - Ipopt::WsmpSolverInterface, 911
- IPOPT_EXPORT
 - IpIpoptApplication.hpp, 996
 - IpStdCInterface.h, 1001, 1003
- IPOPT_UNUSED
 - IpSmartPtr.hpp, 974
- IPOPT_VERSION
 - config_ipopt_default.h, 967
- iPosFirst
 - Ipopt::TripletToCSRConverter, 873
- IRow
 - Ipopt::TripletToCSRConverter::TripletEntry, 865
- iRows_
 - Ipopt::GenTMatrixSpace, 271
- Ipopt::SymTMatrixSpace, 813
- ia_
 - Ipopt::TripletToCSRConverter, 873
- icntl_
 - Ipopt::Ma27TSolverInterface, 490
- ident_space_ds_
 - Ipopt::StdAugSystemSolver, 764
- IdentityMatrix
 - Ipopt::IdentityMatrix, 278
- IdentityMatrixSpace
 - Ipopt::IdentityMatrixSpace, 281
- ignore_singularity_
 - Ipopt::Ma27TSolverInterface, 490
- ikeep_
 - Ipopt::Ma27TSolverInterface, 490
- in_restoration_phase
 - Ipopt::IpoptCalculatedQuantities, 365
- in_soft_resto_phase_
 - Ipopt::BacktrackingLineSearch, 106
- in_tt2_
 - Ipopt::InexactLSAcceptor, 305
- in_watchdog_
 - Ipopt::BacktrackingLineSearch, 105
- Inc_info_iters_since_header
 - Ipopt::IpoptData, 384
- IncreaseQuality
 - Ipopt::AugRestoSystemSolver, 90
 - Ipopt::AugSystemSolver, 95
 - Ipopt::GenAugSystemSolver, 259
 - Ipopt::GenKKTSolverInterface, 264
 - Ipopt::IterativePardisoSolverInterface, 422
 - Ipopt::IterativeWsmpSolverInterface, 434
 - Ipopt::LowRankAugSystemSolver, 467
 - Ipopt::LowRankSSAugSystemSolver, 473
 - Ipopt::Ma27TSolverInterface, 488
 - Ipopt::Ma57TSolverInterface, 497
 - Ipopt::Ma77SolverInterface, 510
 - Ipopt::Ma86SolverInterface, 519
 - Ipopt::Ma97SolverInterface, 530
 - Ipopt::MumpsSolverInterface, 563
 - Ipopt::PardisoSolverInterface, 627
 - Ipopt::SparseSymLinearSolverInterface, 752
 - Ipopt::StdAugSystemSolver, 763
 - Ipopt::SymLinearSolver, 794
 - Ipopt::TSymLinearSolver, 881
 - Ipopt::WsmpSolverInterface, 908
- Index
 - Ipopt, 44
 - IpStdCInterface.h, 1001
- index
 - ma77_info_d, 506
- Index_Type
 - Ipopt::AmplSuffixHandler, 76
- index_g_skip_

- Ipopt::TNLPReducer, [857](#)
- index_style
 - IpStdCInterface.h, [1005](#)
- index_style_
 - Ipopt::StdInterfaceTNLP, [773](#)
 - Ipopt::TNLPAdapter, [848](#)
- index_style_orig_
 - Ipopt::TNLPReducer, [857](#)
- index_x_fix_
 - Ipopt::TNLPReducer, [858](#)
- index_xL_skip_
 - Ipopt::TNLPReducer, [858](#)
- index_xU_skip_
 - Ipopt::TNLPReducer, [858](#)
- IndexStyleEnum
 - Ipopt::TNLP, [832](#)
- InexactCq
 - Ipopt::InexactLSAcceptor, [302](#)
 - Ipopt::InexactNewtonNormalStep, [307](#)
 - Ipopt::InexactNormalStepCalculator, [309](#)
 - Ipopt::InexactPDSolver, [316](#)
 - Ipopt::InexactSearchDirCalculator, [326](#)
 - Ipopt::InexactTSymScalingMethod, [328](#)
 - Ipopt::IterativePardisoSolverInterface, [423](#)
 - Ipopt::IterativeSolverTerminationTester, [430](#)
- InexactData
 - Ipopt::InexactCq, [288](#)
 - Ipopt::InexactLSAcceptor, [302](#)
 - Ipopt::InexactNewtonNormalStep, [307](#)
 - Ipopt::InexactNormalStepCalculator, [309](#)
 - Ipopt::InexactPDSolver, [316](#)
 - Ipopt::InexactSearchDirCalculator, [326](#)
 - Ipopt::IterativePardisoSolverInterface, [423](#)
 - Ipopt::IterativeSolverTerminationTester, [430](#)
- inexact_algorithm_
 - Ipopt::IpoptApplication, [345](#)
- inexact_decomposition_activate_tol_
 - Ipopt::InexactLSAcceptor, [304](#)
- inexact_decomposition_inactivate_tol_
 - Ipopt::InexactLSAcceptor, [304](#)
- inexact_desired_pd_residual_
 - Ipopt::InexactPDTerminationTester, [321](#)
- inexact_desired_pd_residual_iter_
 - Ipopt::InexactPDTerminationTester, [321](#)
- inexact_normal_max_iter_
 - Ipopt::InexactNormalTerminationTester, [313](#)
- inexact_normal_tol_
 - Ipopt::InexactNormalTerminationTester, [313](#)
- inexact_pd_solver_
 - Ipopt::InexactSearchDirCalculator, [326](#)
- inexact_regularization_ls_count_trigger_
 - Ipopt::InexactPDSolver, [317](#)
- InexactAlgorithmBuilder
 - Ipopt::InexactAlgorithmBuilder, [283](#)
- InexactCq
 - Ipopt::InexactCq, [286](#)
- InexactData
 - Ipopt::InexactData, [291](#)
- InexactDoglegNormalStep
 - Ipopt::InexactDoglegNormalStep, [295](#)
- InexactLSAcceptor
 - Ipopt::InexactLSAcceptor, [300](#)
- InexactNewtonNormalStep
 - Ipopt::InexactNewtonNormalStep, [306](#), [307](#)
- InexactNormalStepCalculator
 - Ipopt::InexactNormalStepCalculator, [309](#)
- InexactNormalTerminationTester
 - Ipopt::InexactNormalTerminationTester, [311](#)
- InexactPDSolver
 - Ipopt::InexactPDSolver, [315](#)
- InexactPDTerminationTester
 - Ipopt::InexactPDTerminationTester, [319](#)
- InexactSearchDirCalculator
 - Ipopt::InexactSearchDirCalculator, [325](#)
- InexactTSymScalingMethod
 - Ipopt::InexactTSymScalingMethod, [328](#)
- inf_pr_output_
 - Ipopt::OrigIterationOutput, [623](#)
 - Ipopt::RestIterationOutput, [717](#)
- InfPrOutput
 - Ipopt::IterationOutput, [416](#)
- infeasi
 - Ipopt::PiecewisePenEntry, [658](#)
- Infeasibilities
 - Ipopt::SolveStatistics, [745](#)
- Infeasible_Problem_Detected
 - IpReturnCodes_inc.h, [997](#)
 - Ipopt, [48](#)
- infnorm
 - ma77_control_d, [502](#)
- info_alpha_dual
 - Ipopt::IpoptData, [384](#)
- info_alpha_dual_
 - Ipopt::IpoptData, [388](#)
- info_alpha_primal
 - Ipopt::IpoptData, [383](#)
- info_alpha_primal_
 - Ipopt::IpoptData, [387](#)
- info_alpha_primal_char
 - Ipopt::IpoptData, [383](#)
- info_alpha_primal_char_
 - Ipopt::IpoptData, [387](#)
- info_iters_since_header
 - Ipopt::IpoptData, [384](#)
- info_iters_since_header_
 - Ipopt::IpoptData, [388](#)
- info_last_output
 - Ipopt::IpoptData, [384](#)

- info_last_output_
 - Ipopt::IpoptData, 388
- info_ls_count
 - Ipopt::IpoptData, 384
- info_ls_count_
 - Ipopt::IpoptData, 388
- info_regu_x
 - Ipopt::IpoptData, 383
- info_regu_x_
 - Ipopt::IpoptData, 387
- info_skip_output
 - Ipopt::IpoptData, 384
- info_skip_output_
 - Ipopt::IpoptData, 388
- info_string
 - Ipopt::IpoptData, 384
- info_string_
 - Ipopt::IpoptData, 388
- init_dual_inf_
 - Ipopt::AdaptiveMuUpdate, 62
- init_primal_inf_
 - Ipopt::AdaptiveMuUpdate, 62
- InitPiecewisePenaltyList
 - Ipopt::PiecewisePenalty, 657
- InitThisLineSearch
 - Ipopt::BacktrackingLSAceptor, 109
 - Ipopt::CGPenaltyLSAceptor, 132
 - Ipopt::FilterLSAceptor, 251
 - Ipopt::InexactLSAceptor, 300
 - Ipopt::PenaltyLSAceptor, 651
- Initialize
 - Ipopt::AlgorithmStrategyObject, 68
 - Ipopt::CGPenaltyCq, 119
 - Ipopt::CGPenaltyData, 125
 - Ipopt::InexactCq, 286
 - Ipopt::InexactData, 291
 - Ipopt::IpoptAdditionalCq, 330
 - Ipopt::IpoptAdditionalData, 332
 - Ipopt::IpoptApplication, 341, 342
 - Ipopt::IpoptCalculatedQuantities, 356
 - Ipopt::IpoptData, 380
 - Ipopt::IpoptNLP, 394
 - Ipopt::NLPScalingObject, 584
 - Ipopt::OrigIpoptNLP, 610
 - Ipopt::RestoIpoptNLP, 704
- initialize_called_
 - Ipopt::AlgorithmStrategyObject, 69
 - Ipopt::CGPenaltyCq, 122
 - Ipopt::CGPenaltyData, 128
 - Ipopt::IpoptCalculatedQuantities, 374
 - Ipopt::IpoptData, 387
- initialize_findiff_jac
 - Ipopt::TNLPAdapter, 845
- InitializeConverter
 - Ipopt::TripletToCSRConverter, 873
- InitializeDataStructures
 - Ipopt::CGPenaltyData, 125
 - Ipopt::InexactData, 291
 - Ipopt::IpoptAdditionalData, 332
 - Ipopt::IpoptData, 380
- InitializeFixedMuGlobalization
 - Ipopt::AdaptiveMuUpdate, 59
- InitializeImpl
 - Ipopt::AdaptiveMuUpdate, 59
 - Ipopt::AlgorithmStrategyObject, 68
 - Ipopt::AugRestoSystemSolver, 90
 - Ipopt::AugSystemSolver, 94
 - Ipopt::BacktrackingLineSearch, 100
 - Ipopt::BacktrackingLSAceptor, 109
 - Ipopt::CGPenaltyLSAceptor, 132
 - Ipopt::CGPerturbationHandler, 142
 - Ipopt::CGSearchDirCalculator, 149
 - Ipopt::ConvergenceCheck, 182
 - Ipopt::DefaultIterateInitializer, 185
 - Ipopt::EqMultiplierCalculator, 223
 - Ipopt::EquilibrationScaling, 225
 - Ipopt::ExactHessianUpdater, 227
 - Ipopt::FilterLSAceptor, 251
 - Ipopt::GenAugSystemSolver, 259
 - Ipopt::GenKKTSolverInterface, 263
 - Ipopt::GradientScaling, 274
 - Ipopt::HessianUpdater, 276
 - Ipopt::InexactDoglegNormalStep, 296
 - Ipopt::InexactLSAceptor, 300
 - Ipopt::InexactNewtonNormalStep, 307
 - Ipopt::InexactNormalStepCalculator, 309
 - Ipopt::InexactNormalTerminationTester, 312
 - Ipopt::InexactPDSolver, 315
 - Ipopt::InexactPDTerminationTester, 320
 - Ipopt::InexactSearchDirCalculator, 326
 - Ipopt::InexactTSymScalingMethod, 328
 - Ipopt::IpoptAlgorithm, 335
 - Ipopt::IterateInitializer, 400
 - Ipopt::IterationOutput, 417
 - Ipopt::IterativePardisoSolverInterface, 422
 - Ipopt::IterativeSolverTerminationTester, 429
 - Ipopt::IterativeWsmpSolverInterface, 433
 - Ipopt::LeastSquareMultipliers, 445
 - Ipopt::LimMemQuasiNewtonUpdater, 451
 - Ipopt::LoqoMuOracle, 463
 - Ipopt::LowRankAugSystemSolver, 466
 - Ipopt::LowRankSSAugSystemSolver, 473
 - Ipopt::Ma27TSolverInterface, 487
 - Ipopt::Ma28TDependencyDetector, 493
 - Ipopt::Ma57TSolverInterface, 497
 - Ipopt::Ma77SolverInterface, 509
 - Ipopt::Ma86SolverInterface, 518
 - Ipopt::Ma97SolverInterface, 529

- Ipopt::Mc19TSymScalingMethod, 543
- Ipopt::MinC_1NrmRestorationPhase, 548
- Ipopt::MonotoneMuUpdate, 552
- Ipopt::MumpsSolverInterface, 563
- Ipopt::MuOracle, 568
- Ipopt::MuUpdate, 569
- Ipopt::NLPScalingObject, 587
- Ipopt::OptimalityErrorConvergenceCheck, 595
- Ipopt::OrigIterationOutput, 622
- Ipopt::PardisoSolverInterface, 627
- Ipopt::PDFullSpaceSolver, 633
- Ipopt::PDPerturbationHandler, 639
- Ipopt::PDSearchDirCalculator, 644
- Ipopt::PDSysSystemSolver, 647
- Ipopt::PenaltyLSAcceptor, 651
- Ipopt::ProbingMuOracle, 664
- Ipopt::QualityFunctionMuOracle, 669
- Ipopt::RestoConvergenceCheck, 695
- Ipopt::RestoFilterConvergenceCheck, 698
- Ipopt::RestoIterateInitializer, 714
- Ipopt::RestoIterationOutput, 717
- Ipopt::RestoPenaltyConvergenceCheck, 720
- Ipopt::RestorationPhase, 722
- Ipopt::RestoRestorationPhase, 724
- Ipopt::SearchDirectionCalculator, 733
- Ipopt::SlackBasedTSymScalingMethod, 735
- Ipopt::SparseSymLinearSolverInterface, 751
- Ipopt::StandardScalingBase, 758
- Ipopt::StdAugSystemSolver, 762
- Ipopt::SymLinearSolver, 794
- Ipopt::TDependencyDetector, 818
- Ipopt::TSymDependencyDetector, 876
- Ipopt::TSymLinearSolver, 880
- Ipopt::TSymScalingMethod, 884
- Ipopt::WarmStartIterateInitializer, 903
- Ipopt::WsmpSolverInterface, 908
- InitializeIterates
 - Ipopt::IpoptAlgorithm, 336
 - Ipopt::TimingStatistics, 825
- InitializeIterates_
 - Ipopt::TimingStatistics, 827
- InitializeSolve
 - Ipopt::InexactNormalTerminationTester, 312
 - Ipopt::InexactPDTerminationTester, 320
 - Ipopt::IterativeSolverTerminationTester, 429
- InitializeStructure
 - Ipopt::IterativePardisoSolverInterface, 422
 - Ipopt::IterativeWsmpSolverInterface, 433
 - Ipopt::Ma27TSolverInterface, 487
 - Ipopt::Ma57TSolverInterface, 497
 - Ipopt::Ma77SolverInterface, 510
 - Ipopt::Ma86SolverInterface, 518
 - Ipopt::Ma97SolverInterface, 529
 - Ipopt::MumpsSolverInterface, 563
 - Ipopt::PardisoSolverInterface, 627
 - Ipopt::SparseSymLinearSolverInterface, 751
 - Ipopt::TSymLinearSolver, 881
 - Ipopt::WsmpSolverInterface, 908
- InitializeStructures
 - Ipopt::IpoptNLP, 395
 - Ipopt::OrigIpoptNLP, 610
 - Ipopt::RestoIpoptNLP, 704
- initialized_
 - Ipopt::DenseGenMatrix, 193
 - Ipopt::DenseSymMatrix, 199
 - Ipopt::DenseVector, 209
 - Ipopt::GenTMatrix, 269
 - Ipopt::IterativePardisoSolverInterface, 426
 - Ipopt::IterativeWsmpSolverInterface, 436
 - Ipopt::Ma27TSolverInterface, 489
 - Ipopt::Ma57TSolverInterface, 499
 - Ipopt::MonotoneMuUpdate, 553
 - Ipopt::MumpsSolverInterface, 565
 - Ipopt::OptionsList::OptionValue, 604
 - Ipopt::OrigIpoptNLP, 620
 - Ipopt::PardisoSolverInterface, 629
 - Ipopt::QualityFunctionMuOracle, 672
 - Ipopt::RestoIpoptNLP, 712
 - Ipopt::SymTMatrix, 810
 - Ipopt::TripletToCSRConverter, 874
 - Ipopt::TSymLinearSolver, 882
 - Ipopt::WsmpSolverInterface, 911
- Insufficient_Memory
 - IpReturnCodes_inc.h, 998
 - Ipopt, 49
- Int
 - Ipopt, 45
 - IpStdCInterface.h, 1001
- Integer_Option
 - Ipopt::AmplOptionsList, 73
- integer_meta_data_
 - Ipopt::DenseVectorSpace, 213
- IntegerMetaDataMapType
 - Ipopt, 45
 - Ipopt::TNLP, 831
- Interfaces/lpAlgTypes.hpp, 994
- Interfaces/lpInterfacesRegOp.hpp, 995
- Interfaces/lpIpoptApplication.hpp, 995
- Interfaces/lpNLP.hpp, 996
- Interfaces/lpReturnCodes.h, 996
- Interfaces/lpReturnCodes.hpp, 996
- Interfaces/lpReturnCodes_inc.h, 997
- Interfaces/lpSolveStatistics.hpp, 998
- Interfaces/lpStdCInterface.h, 998
- Interfaces/lpStdInterfaceTNLP.hpp, 1007
- Interfaces/lpTNLP.hpp, 1007
- Interfaces/lpTNLPAdapter.hpp, 1008
- Interfaces/lpTNLPReducer.hpp, 1008

- Intermediate_CB
 - IpStdCInterface.h, [1003](#)
- intermediate_callback
 - Ipopt::StdInterfaceTNLP, [771](#)
 - Ipopt::TNLP, [835](#)
 - Ipopt::TNLPReducer, [856](#)
- intermediate_cb
 - IpStdCInterface.h, [1006](#)
- intermediate_cb_
 - Ipopt::StdInterfaceTNLP, [774](#)
- IntermediateCallback
 - Ipopt::IpoptNLP, [398](#)
 - Ipopt::NLP, [574](#)
 - Ipopt::NLPBoundsRemover, [579](#)
 - Ipopt::OrigIpoptNLP, [614](#)
 - Ipopt::RestIpoptNLP, [707](#)
 - Ipopt::TNLPAdapter, [844](#)
- Internal_Error
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- internal_conval
 - Ipopt::AmplTNLP, [85](#)
- internal_eval_g
 - Ipopt::TNLPAdapter, [845](#)
- internal_eval_jac_g
 - Ipopt::TNLPAdapter, [845](#)
- internal_objval
 - Ipopt::AmplTNLP, [85](#)
- InternalSymFact
 - Ipopt::IterativeWsmvSolverInterface, [434](#)
 - Ipopt::WsmvSolverInterface, [909](#)
- Invalid_Number_Detected
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- Invalid_Option
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- Invalid_Problem_Definition
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- Invalidate
 - Ipopt::DependentResult, [215](#)
- InvalidateResult
 - Ipopt::CachedResults, [116](#)
- iostat
 - ma77_info_d, [504](#)
 - mc68_info, [545](#)
- IpReturnCodes_inc.h
 - Diverging_Iterates, [997](#)
 - Error_In_Step_Computation, [998](#)
 - Feasible_Point_Found, [998](#)
 - Infeasible_Problem_Detected, [997](#)
 - Insufficient_Memory, [998](#)
 - Internal_Error, [998](#)
 - Invalid_Number_Detected, [998](#)
 - Invalid_Option, [998](#)
 - Invalid_Problem_Definition, [998](#)
 - Maximum_CpuTime_Exceeded, [998](#)
 - Maximum_Iterations_Exceeded, [998](#)
 - NonIpopt_Exception_Thrown, [998](#)
 - Not_Enough_Degrees_Of_Freedom, [998](#)
 - RegularMode, [998](#)
 - Restoration_Failed, [998](#)
 - RestorationPhaseMode, [998](#)
 - Search_Direction_Becomes_Too_Small, [997](#)
 - Solve_Succeeded, [997](#)
 - Solved_To_Acceptable_Level, [997](#)
 - Unrecoverable_Exception, [998](#)
 - User_Requested_Stop, [998](#)
- ip_cq_
 - Ipopt::AlgorithmStrategyObject, [69](#)
 - Ipopt::CGPenaltyCq, [121](#)
 - Ipopt::InexactCq, [288](#)
 - Ipopt::IpoptApplication, [345](#)
- ip_data_
 - Ipopt::AlgorithmStrategyObject, [69](#)
 - Ipopt::CGPenaltyCq, [121](#)
 - Ipopt::InexactCq, [288](#)
 - Ipopt::IpoptApplication, [344](#)
 - Ipopt::IpoptCalculatedQuantities, [365](#)
- ip_nlp_
 - Ipopt::AlgorithmStrategyObject, [69](#)
 - Ipopt::CGPenaltyCq, [121](#)
 - Ipopt::InexactCq, [288](#)
 - Ipopt::IpoptApplication, [344](#)
 - Ipopt::IpoptCalculatedQuantities, [365](#)
- IpBlasDasum
 - Ipopt, [53](#)
- IpBlasDaxpy
 - Ipopt, [54](#)
- IpBlasDcopy
 - Ipopt, [54](#)
- IpBlasDdot
 - Ipopt, [53](#)
- IpBlasDgemm
 - Ipopt, [54](#)
- IpBlasDgemv
 - Ipopt, [54](#)
- IpBlasDnrm2
 - Ipopt, [53](#)
- IpBlasDscal
 - Ipopt, [54](#)
- IpBlasDsylv
 - Ipopt, [54](#)
- IpBlasDsyrk
 - Ipopt, [54](#)
- IpBlasDtrsm
 - Ipopt, [54](#)

IpBlasIdamax
 Ipopt, 53
 IpCq
 Ipopt::AlgorithmStrategyObject, 69
 IpData
 Ipopt::AlgorithmStrategyObject, 69
 IpDebug.hpp
 DBG_ASSERT, 968
 DBG_DO, 968
 DBG_EXEC, 968
 DBG_PRINT, 968
 DBG_PRINT_MATRIX, 968
 DBG_PRINT_VECTOR, 968
 DBG_START_FUN, 968
 DBG_START_METH, 968
 DBG_VERBOSITY, 968
 IpException.hpp
 ASSERT_EXCEPTION, 969
 THROW_EXCEPTION, 969
 IpIpoptApplication.hpp
 IPOPT_EXPORT, 996
 IpLapackDgetrf
 Ipopt, 55
 IpLapackDgetrs
 Ipopt, 55
 IpLapackDpotrf
 Ipopt, 55
 IpLapackDpotrs
 Ipopt, 54
 IpLapackDsyeve
 Ipopt, 55
 IpMa57TSolverInterface.hpp
 ma57int, 960
 IpMatrix.hpp
 DBG_PRINT_MATRIX, 1015
 IpNLP
 Ipopt::AlgorithmStrategyObject, 69
 IpRandom01
 Ipopt, 52
 IpResetRandom01
 Ipopt, 52
 IpReturnCodes_inc.h
 AlgorithmMode, 998
 ApplicationReturnStatus, 997
 IpSmartPtr.hpp
 IPOPT_UNUSED, 974
 ipopt_dbg_smartptr_verbosity, 974
 IpStdClInterface.h
 Bool, 1002
 Eval_F_CB, 1002
 Eval_G_CB, 1002
 Eval_Grad_F_CB, 1002
 Eval_H_CB, 1002
 Eval_Jac_G_CB, 1002
 eval_f, 1005
 eval_g, 1005
 eval_grad_f, 1005
 eval_h, 1005
 eval_jac_g, 1005
 FALSE, 1001
 file_name, 1005
 g, 1006
 g_L, 1004
 g_U, 1004
 g_scaling, 1006
 IPOPT_EXPORT, 1001, 1003
 Index, 1001
 index_style, 1005
 Int, 1001
 Intermediate_CB, 1003
 intermediate_cb, 1006
 IpoptProblem, 1002
 keyword, 1005
 m, 1004
 mult_g, 1006
 mult_x_L, 1006
 mult_x_U, 1006
 nele_hess, 1004
 nele_jac, 1004
 Number, 1001
 obj_scaling, 1005
 obj_val, 1006
 print_level, 1005
 TRUE, 1001
 user_data, 1006
 UserDataPtr, 1002
 val, 1005
 x, 1006
 x_L, 1004
 x_U, 1004
 x_scaling, 1006
 IpTypes.hpp
 ipfint, 975
 IpVector.hpp
 DBG_PRINT_VECTOR, 1018
 ipfint
 HSLLoader.h, 986
 IpTypes.hpp, 975
 Ipopt, 32
 ALL_VARS, 46
 AddInexactDefaultOptions, 49
 AlgorithmMode, 49
 ApplicationReturnStatus, 48
 CPUTIME_EXCEEDED, 48
 Compare_le, 52
 ComparePointers, 51
 ConstPtr, 50
 CpuTime, 52

DIVERGING_ITERATES, [48](#)
 Diverging_Iterates, [48](#)
 ERROR_IN_STEP_COMPUTATION, [48](#)
 EXACT, [45](#)
 EJournalCategory, [46](#)
 EJournalLevel, [46](#)
 ENormType, [45](#)
 ESymSolverStatus, [46](#)
 Error_In_Step_Computation, [49](#)
 FEASIBLE_POINT_FOUND, [48](#)
 Feasible_Point_Found, [48](#)
 GetRawPtr, [50](#)
 HessianApproximationSpace, [45](#)
 HessianApproximationType, [45](#)
 INTERNAL_ERROR, [48](#)
 INVALID_NUMBER_DETECTED, [48](#)
 INVALID_OPTION, [48](#)
 Index, [44](#)
 Infeasible_Problem_Detected, [48](#)
 Insufficient_Memory, [49](#)
 Int, [45](#)
 IntegerMetaDataMapType, [45](#)
 Internal_Error, [49](#)
 Invalid_Number_Detected, [49](#)
 Invalid_Option, [49](#)
 Invalid_Problem_Definition, [49](#)
 IpBlasDasum, [53](#)
 IpBlasDaxpy, [54](#)
 IpBlasDcopy, [54](#)
 IpBlasDdot, [53](#)
 IpBlasDgemm, [54](#)
 IpBlasDgemv, [54](#)
 IpBlasDnrm2, [53](#)
 IpBlasDscal, [54](#)
 IpBlasDsylv, [54](#)
 IpBlasDsyrk, [54](#)
 IpBlasDtrsm, [54](#)
 IpBlasIdamax, [53](#)
 IpLapackDgetrf, [55](#)
 IpLapackDgetrs, [55](#)
 IpLapackDpotrf, [55](#)
 IpLapackDpotrs, [54](#)
 IpLapackDsylv, [55](#)
 IpRandom01, [52](#)
 IpResetRandom01, [52](#)
 IsFiniteNumber, [52](#)
 IsNull, [50](#)
 IsValid, [50](#)
 J_ALL, [46](#)
 J_BARRIER_UPDATE, [47](#)
 J_DBG, [47](#)
 J_DETAILED, [46](#)
 J_DOCUMENTATION, [47](#)
 J_ERROR, [46](#)
 J_FRAC_TO_BOUND, [47](#)
 J_HESSIAN_APPROXIMATION, [47](#)
 J_INITIALIZATION, [47](#)
 J_INSUPPRESSIBLE, [46](#)
 J_ITERSUMMARY, [46](#)
 J_LAST_CATEGORY, [47](#)
 J_LAST_LEVEL, [46](#)
 J_LINE_SEARCH, [47](#)
 J_LINEAR_ALGEBRA, [47](#)
 J_MAIN, [47](#)
 J_MATRIX, [46](#)
 J_MOREDETAILED, [46](#)
 J_MOREMATRIX, [46](#)
 J_MOREVECTOR, [46](#)
 J_NLP, [47](#)
 J_NONE, [46](#)
 J_SOLUTION, [47](#)
 J_SOLVE_PD_SYSTEM, [47](#)
 J_STATISTICS, [47](#)
 J_STRONGWARNING, [46](#)
 J_SUMMARY, [46](#)
 J_TIMING_STATISTICS, [47](#)
 J_USER1, [47](#)
 J_USER10, [47](#)
 J_USER11, [47](#)
 J_USER12, [47](#)
 J_USER13, [47](#)
 J_USER14, [47](#)
 J_USER15, [47](#)
 J_USER16, [47](#)
 J_USER17, [47](#)
 J_USER2, [47](#)
 J_USER3, [47](#)
 J_USER4, [47](#)
 J_USER5, [47](#)
 J_USER6, [47](#)
 J_USER7, [47](#)
 J_USER8, [47](#)
 J_USER9, [47](#)
 J_USER_APPLICATION, [47](#)
 J_VECTOR, [46](#)
 J_WARNING, [46](#)
 LIMITED_MEMORY, [45](#)
 LOCAL_INFEASIBILITY, [48](#)
 MAXITER_EXCEEDED, [48](#)
 Max, [51](#), [52](#)
 Maximum_CpuTime_Exceeded, [49](#)
 Maximum_Iterations_Exceeded, [49](#)
 Min, [51](#), [52](#)
 NONLINEAR_VARS, [46](#)
 NORM_1, [45](#)
 NORM_2, [45](#)
 NORM_MAX, [45](#)
 NonIpopt_Exception_Thrown, [49](#)

- Not_Enough_Degrees_Of_Freedom, 49
- Number, 44
- NumericMetaDataMapType, 45
- OT_Integer, 48
- OT_Number, 48
- OT_String, 48
- OT_Unknown, 48
- OUT_OF_MEMORY, 48
- operator<, 51
- operator<=, 51
- operator>, 51
- operator>=, 51
- operator+, 51
- operator==, 50
- PiecewisePenEntry, 45
- RESTORATION_FAILURE, 48
- RegisterOptions_Algorithm, 49
- RegisterOptions_CGPenalty, 53
- RegisterOptions_Inexact, 49
- RegisterOptions_Interfaces, 53
- RegisterOptions_LinearSolvers, 50
- RegisteredOptionType, 47
- RegularMode, 49
- Restoration_Failed, 49
- RestorationPhaseMode, 49
- STOP_AT_ACCEPTABLE_POINT, 48
- STOP_AT_TINY_STEP, 48
- SUCCESS, 48
- SYMSOLVER_CALL_AGAIN, 46
- SYMSOLVER_FATAL_ERROR, 46
- SYMSOLVER_SINGULAR, 46
- SYMSOLVER_SUCCESS, 46
- SYMSOLVER_WRONG_INERTIA, 46
- Search_Direction_Becomes_Too_Small, 48
- Snprintf, 52
- Solve_Succeeded, 48
- Solved_To_Acceptable_Level, 48
- SolverReturn, 48
- StringMetaDataMapType, 45
- swap, 51
- SysTime, 52
- TOO_FEW_DEGREES_OF_FREEDOM, 48
- UNASSIGNED, 48
- USER_REQUESTED_STOP, 48
- Unrecoverable_Exception, 49
- User_Requested_Stop, 48
- WallclockTime, 52
- Ipopt::AdaptiveMuUpdate
 - FILTER_OBJ_CONSTR, 58
 - KKT_ERROR, 58
 - NEVER_MONOTONE_MODE, 59
- Ipopt::AmplOptionsList
 - HaltOnError_Option, 73
 - Integer_Option, 73
 - Number_Option, 73
 - String_Option, 73
 - WS_Option, 73
- Ipopt::AmplSuffixHandler
 - Constraint_Source, 76
 - Index_Type, 76
 - Number_Type, 76
 - Objective_Source, 76
 - Problem_Source, 76
 - Variable_Source, 76
- Ipopt::BacktrackingLineSearch
 - DUAL_ALPHA_FOR_Y, 100
 - DUAL_AND_FULL_ALPHA_FOR_Y, 100
 - FULL_STEP_FOR_Y, 100
 - LSACCEPTOR_ALPHA_FOR_Y, 100
 - MAX_ALPHA_FOR_Y, 100
 - MIN_ALPHA_FOR_Y, 100
 - MIN_DUAL_INFEAS_ALPHA_FOR_Y, 100
 - PRIMAL_ALPHA_FOR_Y, 100
 - PRIMAL_AND_FULL_ALPHA_FOR_Y, 100
 - SAFE_MIN_DUAL_INFEAS_ALPHA_FOR_Y, 100
- Ipopt::CGPerturbationHandler
 - DEGENERATE, 142
 - NO_TEST, 142
 - NOT_DEGENERATE, 142
 - NOT_YET_DETERMINED, 142
 - TEST_DELTA_C_EQ_0_DELTA_X_EQ_0, 142
 - TEST_DELTA_C_EQ_0_DELTA_X_GT_0, 142
 - TEST_DELTA_C_GT_0_DELTA_X_EQ_0, 142
 - TEST_DELTA_C_GT_0_DELTA_X_GT_0, 142
- Ipopt::ConvergenceCheck
 - CONTINUE, 181
 - CONVERGED, 181
 - CONVERGED_TO_ACCEPTABLE_POINT, 181
 - CPUTIME_EXCEEDED, 181
 - DIVERGING, 181
 - FAILED, 181
 - MAXITER_EXCEEDED, 181
 - USER_STOP, 181
- Ipopt::DefaultIterateInitializer
 - B_CONSTANT, 185
 - B_MU_BASED, 185
- Ipopt::DenseGenMatrix
 - CHOL, 190
 - LU, 190
 - NONE, 190
- Ipopt::FilterLSAcceptor
 - AFFINE_CORRECTOR, 250
 - NO_CORRECTOR, 250
 - PRIMAL_DUAL_CORRECTOR, 250
- Ipopt::InexactSearchDirCalculator
 - ADAPTIVE, 325
 - ALWAYS, 325
 - SWITCH_ONCE, 325

- Ipopt::IterationOutput
 - INTERNAL, 416
 - ORIGINAL, 416
- Ipopt::IterativePardisoSolverInterface
 - COMPLETE, 421
 - COMPLETE2x2, 421
 - CONSTRAINT, 421
- Ipopt::IterativeSolverTerminationTester
 - CONTINUE, 429
 - MODIFY_HESSIAN, 429
 - OTHER_SATISFIED, 429
 - TEST_1_SATISFIED, 429
 - TEST_2_SATISFIED, 429
 - TEST_3_SATISFIED, 429
- Ipopt::LimMemQuasiNewtonUpdater
 - BFGS, 450
 - CONSTANT, 451
 - SCALAR1, 451
 - SCALAR2, 451
 - SCALAR3, 451
 - SCALAR4, 451
 - SR1, 450
- Ipopt::Ma77SolverInterface
 - ORDER_AMD, 509
 - ORDER_METIS, 509
- Ipopt::Ma86SolverInterface
 - ORDER_AMD, 518
 - ORDER_AUTO, 518
 - ORDER_METIS, 518
- Ipopt::Ma97SolverInterface
 - ORDER_AMD, 528
 - ORDER_AUTO, 528
 - ORDER_BEST, 528
 - ORDER_MATCHED_AMD, 528
 - ORDER_MATCHED_AUTO, 528
 - ORDER_MATCHED_METIS, 528
 - ORDER_METIS, 528
 - SWITCH_AT_START, 528
 - SWITCH_AT_START_REUSE, 528
 - SWITCH_NDELAY, 528
 - SWITCH_NDELAY_REUSE, 528
 - SWITCH_NEVER, 528
 - SWITCH_OD_ND, 528
 - SWITCH_OD_ND_REUSE, 528
 - SWITCH_ON_DEMAND, 528
 - SWITCH_ON_DEMAND_REUSE, 528
- Ipopt::Observer
 - NT_All, 591
 - NT_BeingDestroyed, 591
 - NT_Changed, 591
- Ipopt::PDPerturbationHandler
 - DEGENERATE, 638
 - NO_TEST, 638
 - NOT_DEGENERATE, 638
 - NOT_YET_DETERMINED, 638
 - TEST_DELTA_C_EQ_0_DELTA_X_EQ_0, 638
 - TEST_DELTA_C_EQ_0_DELTA_X_GT_0, 638
 - TEST_DELTA_C_GT_0_DELTA_X_EQ_0, 638
 - TEST_DELTA_C_GT_0_DELTA_X_GT_0, 639
- Ipopt::PardisoSolverInterface
 - COMPLETE, 626
 - COMPLETE2x2, 626
 - CONSTRAINT, 627
- Ipopt::QualityFunctionMuOracle
 - BT_CUBIC, 668
 - BT_NONE, 668
 - CEN_CUBED_RECIPROCAL, 668
 - CEN_LOG, 668
 - CEN_NONE, 668
 - CEN_RECIPROCAL, 668
 - NM_NORM_1, 668
 - NM_NORM_2, 668
 - NM_NORM_2_SQUARED, 668
 - NM_NORM_MAX, 668
- Ipopt::SparseSymLinearSolverInterface
 - CSR_Format_0_Offset, 750
 - CSR_Format_1_Offset, 750
 - CSR_Full_Format_0_Offset, 750
 - CSR_Full_Format_1_Offset, 750
 - Triplet_Format, 750
- Ipopt::TNLP
 - C_STYLE, 832
 - FORTTRAN_STYLE, 832
 - LINEAR, 832
 - NON_LINEAR, 832
- Ipopt::TNLPAdapter
 - FIRST_ORDER_TEST, 842
 - JAC_EXACT, 842
 - JAC_FINDIFF_VALUES, 842
 - MAKE_CONSTRAINT, 841
 - MAKE_PARAMETER, 841
 - NO_TEST, 842
 - ONLY_SECOND_ORDER_TEST, 842
 - RELAX_BOUNDS, 841
 - SECOND_ORDER_TEST, 842
- Ipopt::TripletToCSRConverter
 - Full_Format, 872
 - Triangular_Format, 872
- Ipopt::AdaptiveMuUpdate, 55
 - ~AdaptiveMuUpdate, 59
 - accepted_point_, 63
 - adaptive_mu_globalization_, 61
 - adaptive_mu_kkt_balancing_term_, 61
 - adaptive_mu_kkt_centrality_, 61
 - adaptive_mu_kkt_norm_, 61
 - adaptive_mu_monotone_init_factor_, 61
 - adaptive_mu_safeguard_factor_, 61
 - AdaptiveMuGlobalizationEnum, 58

- AdaptiveMuUpdate, 59
- barrier_tol_factor_, 61
- check_if_no_bounds_, 63
- CheckSufficientProgress, 59
- compl_inf_tol_, 61
- Compute_tau_monotone, 60
- filter_, 62
- filter_margin_fact_, 61
- filter_max_margin_, 61
- fix_mu_oracle_, 62
- free_mu_oracle_, 62
- init_dual_inf_, 62
- init_primal_inf_, 62
- InitializeFixedMuGlobalization, 59
- InitializImpl, 59
- linesearch_, 62
- lower_mu_safeguard, 60
- max_ref_val, 60
- min_ref_val, 60
- mu_linear_decrease_factor_, 61
- mu_max_, 60
- mu_max_fact_, 60
- mu_min_, 60
- mu_min_default_, 60
- mu_superlinear_decrease_power_, 61
- mu_target_, 60
- NewFixedMu, 60
- no_bounds_, 63
- num_refs_max_, 62
- operator=, 59
- quality_function_pd_system, 60
- refs_red_fact_, 62
- refs_vals_, 62
- RegisterOptions, 59
- RememberCurrentPointAsAccepted, 59
- restore_accepted_iterate_, 62
- tau_min_, 60
- UpdateBarrierParameter, 59
- Ipopt::AlgorithmBuilder, 63
 - ~AlgorithmBuilder, 64
 - AlgorithmBuilder, 64
 - BuildBasicAlgorithm, 65
 - BuildIpoptObjects, 65
 - custom_solver_, 65
 - operator=, 65
 - RegisterOptions, 65
- Ipopt::AlgorithmStrategyObject, 65
 - ~AlgorithmStrategyObject, 67
 - AlgorithmStrategyObject, 67, 68
 - HavelpData, 69
 - Initialize, 68
 - initialize_called_, 69
 - InitializImpl, 68
 - ip_cq_, 69
 - ip_data_, 69
 - ip_nlp_, 69
 - IpCq, 69
 - IpData, 69
 - IpNLP, 69
 - Jnlst, 69
 - jnlst_, 69
 - operator=, 69
 - ReducedInitialize, 68
- Ipopt::AmplOptionsList, 72
 - ~AmplOptionsList, 73
 - AddAmplOption, 74
 - ampl_options_map_, 74
 - AmplOptionType, 73
 - AmplOptionsList, 73
 - keywds_, 74
 - Keywords, 74
 - MakeValidLatexString, 74
 - nkeywds_, 74
 - NumberOfAmplOptions, 74
 - operator=, 74
 - PrintLatex, 74
- Ipopt::AmplOptionsList::AmplOption, 70
 - ~AmplOption, 71
 - AmplOption, 71
 - Description, 71
 - description_, 71
 - ipopt_option_name_, 71
 - IpoptOptionName, 71
 - operator=, 71
 - Type, 71
 - type_, 71
- Ipopt::AmplOptionsList::PrivatInfo, 660
 - ipopt_name_, 661
 - IpoptName, 661
 - Jnlst, 661
 - jnlst_, 662
 - NError, 661
 - nerror_, 662
 - Options, 661
 - options_, 661
 - PrivatInfo, 661
- Ipopt::AmplSuffixHandler, 74
 - ~AmplSuffixHandler, 76
 - AddAvailableSuffix, 76
 - AmplSuffixHandler, 76
 - AmplTNLP, 77
 - asl_, 77
 - GetIntegerSuffixValues, 76
 - GetNumberSuffixValues, 76
 - operator=, 77
 - PrepareAmplForSuffixes, 77
 - Suffix_Source, 76
 - Suffix_Type, 76

- suffix_ids_, 77
- suffix_sources_, 77
- suffix_types_, 77
- suftab_, 77
- Ipopt::AmplTNLP, 77
 - ~AmplTNLP, 81
 - AmplSolverObject, 83
 - AmplTNLP, 81
 - apply_new_x, 85
 - asl_, 85
 - call_heset, 85
 - con_integer_md_, 87
 - con_numeric_md_, 87
 - con_string_md_, 87
 - conval_called_with_current_x_, 86
 - eval_f, 82
 - eval_g, 83
 - eval_grad_f, 82
 - eval_h, 83
 - eval_jac_g, 83
 - finalize_solution, 83
 - g_sol_, 86
 - get_bounds_info, 82
 - get_constraints_linearity, 82
 - get_discrete_info, 84
 - get_list_of_nonlinear_variables, 83
 - get_nlp_info, 82
 - get_number_of_nonlinear_variables, 83
 - get_options, 85
 - get_scaling_parameters, 83
 - get_starting_point, 82
 - get_suffix_handler, 84
 - get_var_con_metadata, 82
 - heset_called_, 86
 - internal_conval, 85
 - internal_objval, 85
 - jnlst_, 85
 - lambda_sol_, 86
 - nerror_, 86
 - nerror_ok, 85
 - nz_h_full_, 85
 - obj_sign_, 85
 - obj_sol_, 86
 - objval_called_with_current_x_, 86
 - Oinfo_ptr_, 86
 - operator=, 84
 - set_active_objective, 84
 - set_active_objective_called_, 86
 - set_integer_metadata_for_con, 84
 - set_integer_metadata_for_var, 84
 - set_numeric_metadata_for_con, 84
 - set_numeric_metadata_for_var, 84
 - set_string_metadata_for_con, 84
 - set_string_metadata_for_var, 84
 - suffix_handler_, 87
 - var_integer_md_, 87
 - var_numeric_md_, 87
 - var_string_md_, 87
 - write_solution_file, 84
 - x_sol_, 85
 - z_L_sol_, 86
 - z_U_sol_, 86
- Ipopt::AugRestoSystemSolver, 87
 - ~AugRestoSystemSolver, 90
 - AugRestoSystemSolver, 89, 90
 - D_x_plus_wr_d, 91
 - d_x_plus_wr_d_cache_, 92
 - IncreaseQuality, 90
 - InitializeImpl, 90
 - Neg_Omega_c_plus_D_c, 91
 - Neg_Omega_d_plus_D_d, 91
 - neg_omega_c_plus_D_c_cache_, 91
 - neg_omega_d_plus_D_d_cache_, 91
 - NumberOfNegEvals, 90
 - operator=, 91
 - orig_aug_solver_, 92
 - ProvidesInertia, 90
 - Rhs_cR, 91
 - rhs_cR_cache_, 92
 - Rhs_dR, 91
 - rhs_dR_cache_, 92
 - Sigma_tilde_n_c_inv, 91
 - sigma_tilde_n_c_inv_cache_, 92
 - Sigma_tilde_n_d_inv, 91
 - sigma_tilde_n_d_inv_cache_, 92
 - Sigma_tilde_p_c_inv, 91
 - sigma_tilde_p_c_inv_cache_, 92
 - Sigma_tilde_p_d_inv, 91
 - sigma_tilde_p_d_inv_cache_, 92
 - skip_orig_aug_solver_init_, 92
 - Solve, 90
- Ipopt::AugSystemSolver, 92
 - ~AugSystemSolver, 94
 - AugSystemSolver, 94
 - IncreaseQuality, 95
 - InitializeImpl, 94
 - MultiSolve, 95
 - NumberOfNegEvals, 95
 - operator=, 95
 - ProvidesInertia, 95
 - Solve, 94
- Ipopt::BacktrackingLSAceptor, 107
 - ~BacktrackingLSAceptor, 108
 - BacktrackingLSAceptor, 108, 109
 - CalculateAlphaMin, 109
 - CheckAcceptabilityOfTrialPoint, 109
 - ComputeAlphaForY, 111
 - DoFallback, 111

- HasComputeAlphaForY, 111
- InitThisLineSearch, 109
- InitializeImpl, 109
- NeverRestorationPhase, 111
- operator=, 111
- PrepareRestoPhaseStart, 109
- RegisterOptions, 111
- Reset, 109
- RestoredIterate, 111
- StartWatchDog, 110
- StopWatchDog, 110
- TryCorrector, 110
- TrySecondOrderCorrection, 110
- UpdateForNextIteration, 110
- Ipopt::BacktrackingLineSearch, 96
 - ~BacktrackingLineSearch, 100
 - accept_after_max_steps_, 104
 - accept_every_trial_step_, 104
 - acceptable_iterate_, 105
 - acceptable_iteration_number_, 106
 - acceptor_, 106
 - ActivateFallbackMechanism, 101
 - alpha_for_y_, 103
 - alpha_for_y_tol_, 103
 - alpha_red_factor_, 103
 - AlphaForYEnum, 100
 - BacktrackingLineSearch, 100
 - CheckAcceptabilityOfTrialPoint, 102
 - CheckSkippedLineSearch, 101
 - conv_check_, 106
 - count_successive_shortened_steps_, 106
 - CurrentIsAcceptable, 103
 - DetectTinyStep, 103
 - DoBacktrackingLineSearch, 101
 - expect_infeasible_problem_, 104
 - expect_infeasible_problem_ctol_, 104
 - expect_infeasible_problem_ytol_, 104
 - fallback_activated_, 106
 - FindAcceptableTrialPoint, 100
 - in_soft_resto_phase_, 106
 - in_watchdog_, 105
 - InitializeImpl, 100
 - last_mu_, 105
 - magic_steps_, 104
 - max_soft_resto_iters_, 103
 - operator=, 101
 - PerformDualStep, 102
 - PerformMagicStep, 102
 - RegisterOptions, 101
 - Reset, 101
 - resto_phase_, 106
 - RestoreAcceptablePoint, 103
 - rigorous_, 106
 - SetRigorousLineSearch, 101
 - skipped_line_search_, 106
 - soft_resto_counter_, 106
 - soft_resto_perror_reduction_factor_, 103
 - start_with_resto_, 105
 - StartWatchDog, 102
 - StopWatchDog, 102
 - StoreAcceptablePoint, 103
 - tiny_step_last_iteration_, 106
 - tiny_step_tol_, 104
 - tiny_step_y_tol_, 104
 - TryCorrector, 102
 - TrySecondOrderCorrection, 102
 - TrySoftRestoStep, 102
 - watchdog_alpha_primal_test_, 105
 - watchdog_delta_, 105
 - watchdog_iterate_, 105
 - watchdog_shortened_iter_, 105
 - watchdog_shortened_iter_trigger_, 105
 - watchdog_trial_iter_, 105
 - watchdog_trial_iter_max_, 105
- Ipopt::CGPenaltyCq, 117
 - ~CGPenaltyCq, 119
 - CGPenData, 120
 - CGPenaltyCq, 119
 - compute_curr_cg_penalty, 120
 - compute_curr_cg_penalty_scale, 120
 - curr_added_y_nrm2, 120
 - curr_added_y_nrm2_cache_, 121
 - curr_cg_pert_fact, 120
 - curr_cg_pert_fact_cache_, 121
 - curr_direct_deriv_penalty_function, 120
 - curr_direct_deriv_penalty_function_cache_, 121
 - curr_fast_direct_deriv_penalty_function, 120
 - curr_fast_direct_deriv_penalty_function_cache_, 121
 - curr_jac_cd_norm, 120
 - curr_jac_cd_norm_cache_, 121
 - curr_penalty_function, 120
 - curr_penalty_function_cache_, 121
 - curr_scaled_y_Amax, 120
 - curr_scaled_y_Amax_cache_, 121
 - dT_times_barH_times_d, 120
 - Initialize, 119
 - initialize_called_, 122
 - ip_cq_, 121
 - ip_data_, 121
 - ip_nlp_, 121
 - operator=, 120
 - reference_infeasibility_, 121
 - RegisterOptions, 120
 - trial_penalty_function, 120
 - trial_penalty_function_cache_, 121
- Ipopt::CGPenaltyData, 122
 - ~CGPenaltyData, 124
 - AcceptTrialPoint, 126

- CGPenaltyData, 124
- curr_kkt_penalty, 127
- curr_kkt_penalty_, 128
- curr_penalty, 126
- curr_penalty_, 128
- curr_penalty_pert_, 128
- CurrPenaltyPert, 126
- delta_cgfast, 125
- delta_cgfast_, 127
- delta_cgpen, 125
- delta_cgpen_, 127
- have_cgfast_deltas_, 127
- have_cgpen_deltas_, 127
- HaveCgFastDeltas, 125
- HaveCgPenDeltas, 125
- Initialize, 125
- initialize_called_, 128
- InitializeDataStructures, 125
- KKTPenaltyInitialized, 127
- kkt_penalty_initialized_, 128
- max_alpha_x_, 128
- never_try_pure_Newton_, 128
- NeverTryPureNewton, 126
- operator=, 127
- penalty_initialized_, 128
- PenaltyInitialized, 127
- PrimalStepSize, 126
- restor_counter, 126
- restor_counter_, 128
- restor_iter, 126
- restor_iter_, 128
- set_delta_cgfast, 125
- set_delta_cgpen, 125
- Set_kkt_penalty, 127
- Set_penalty, 126
- SetCurrPenaltyPert, 126
- SetHaveCgFastDeltas, 126
- SetHaveCgPenDeltas, 125
- SetKKTPenaltyUninitialized, 127
- SetNeverTryPureNewton, 126
- SetPenaltyUninitialized, 127
- SetPrimalStepSize, 126
- SetRestorCounter, 126
- SetRestorIter, 126
- Ipopt::CGPenaltyLSAcceptor, 128
 - ~CGPenaltyLSAcceptor, 132
 - accepted_by_Armijo_, 136
 - ArmijoHolds, 134
 - best_KKT_error_, 137
 - best_iterate_, 137
 - CGPenCq, 134
 - CGPenData, 134
 - CGPenaltyLSAcceptor, 132
 - CalculateAlphaMin, 133
 - CheckAcceptabilityOfTrialPoint, 133
 - chi_cup_, 135
 - chi_hat_, 135
 - chi_tilde_, 135
 - Compare_le, 135
 - counter_first_type_penalty_updates_, 137
 - counter_second_type_penalty_updates_, 137
 - curr_eta_, 137
 - CurrentIsBest, 135
 - DoFallback, 134
 - epsilon_c_, 136
 - eta_min_, 135
 - eta_penalty_, 135
 - gamma_hat_, 135
 - gamma_tilde_, 136
 - InitThisLineSearch, 132
 - InitializeImpl, 132
 - IsAcceptableToPiecewisePenalty, 134
 - jump_for_tiny_step_, 138
 - kappa_soc_, 137
 - Is_counter_, 137
 - max_soc_, 137
 - min_alpha_primal_, 136
 - mult_diverg_feasibility_tol_, 137
 - mult_diverg_y_tol_, 137
 - MultipliersDiverged, 135
 - never_use_piecewise_penalty_Is_, 138
 - NeverRestorationPhase, 134
 - operator=, 134
 - pd_solver_, 138
 - pen_curr_mu_, 136
 - pen_theta_max_, 136
 - pen_theta_max_fact_, 136
 - penalty_max_, 136
 - penalty_update_compl_tol_, 135
 - penalty_update_infeasibility_tol_, 135
 - PiecewisePenalty_, 138
 - piecewisepenalty_gamma_infeasi_, 136
 - piecewisepenalty_gamma_obj_, 136
 - PrepareRestoPhaseStart, 133
 - reference_curr_direct_f_nrm_, 138
 - reference_direct_deriv_penalty_function_, 138
 - reference_penalty_function_, 137
 - reference_theta_, 136
 - RegisterOptions, 134
 - Reset, 132
 - reset_piecewise_penalty_, 138
 - RestoreBestPoint, 135
 - RestoredIterate, 134
 - StartWatchDog, 134
 - StopWatchDog, 134
 - StoreBestPoint, 135
 - theta_min_, 136
 - TryCorrector, 133

- TrySecondOrderCorrection, 133
- UpdateForNextIteration, 133
- UpdatePenaltyParameter, 135
- watchdog_delta_cgpen_, 138
- watchdog_direct_deriv_penalty_function_, 138
- watchdog_penalty_function_, 138
- Ipopt::CGPerturbationHandler, 139
 - ~CGPerturbationHandler, 142
 - CGPenCq, 143
 - CGPenData, 143
 - CGPerturbationHandler, 142
 - ConsiderNewSystem, 142
 - CurrentPerturbation, 143
 - degen_iters_, 145
 - degen_iters_max_, 146
 - DegenType, 142
 - delta_c_curr_, 144
 - delta_c_last_, 144
 - delta_cd, 144
 - delta_cd_exp_, 146
 - delta_cd_val_, 146
 - delta_d_curr_, 144
 - delta_d_last_, 144
 - delta_s_curr_, 144
 - delta_s_last_, 144
 - delta_x_curr_, 144
 - delta_x_last_, 144
 - delta_xs_dec_fact_, 145
 - delta_xs_first_inc_fact_, 145
 - delta_xs_inc_fact_, 145
 - delta_xs_init_, 145
 - delta_xs_max_, 145
 - delta_xs_min_, 145
 - finalize_test, 144
 - get_deltas_for_wrong_inertia, 143
 - get_deltas_for_wrong_inertia_called_, 144
 - hess_degenerate_, 145
 - InitializeImpl, 142
 - jac_degenerate_, 145
 - mult_diverg_feasibility_tol_, 146
 - operator=, 143
 - penalty_max_, 146
 - perturb_always_cd_, 146
 - PerturbForSingularity, 143
 - PerturbForWrongInertia, 143
 - RegisterOptions, 143
 - reset_last_, 146
 - test_status_, 145
 - TrialStatus, 142
- Ipopt::CGSearchDirCalculator, 146
 - ~CGSearchDirCalculator, 148
 - CGPenCq, 149
 - CGPenData, 149
 - CGSearchDirCalculator, 148
 - ComputeSearchDirection, 149
 - delta_y_max_, 150
 - fast_des_fact_, 150
 - InitializeImpl, 149
 - kappa_x_dis_, 150
 - kappa_y_dis_, 150
 - never_use_fact_cgpen_direction_, 150
 - nonmonotone_pen_update_counter_, 150
 - operator=, 149
 - pd_solver_, 150
 - pen_des_fact_, 150
 - pen_init_fac_, 150
 - penalty_backward_, 150
 - penalty_init_max_, 149
 - penalty_init_min_, 149
 - penalty_max_, 149
 - RegisterOptions, 149
 - vartheta_, 150
- Ipopt::CachedResults
 - ~CachedResults, 114
 - AddCachedResult, 114
 - AddCachedResult1Dep, 115, 116
 - AddCachedResult2Dep, 115, 116
 - AddCachedResult3Dep, 115, 116
 - cached_results_, 117
 - CachedResults, 114
 - CleanupInvalidatedResults, 116
 - Clear, 116
 - DebugPrintCachedResults, 116
 - GetCachedResult, 114, 115
 - GetCachedResult1Dep, 115
 - GetCachedResult2Dep, 115
 - GetCachedResult3Dep, 115, 116
 - InvalidateResult, 116
 - max_cache_size_, 117
 - operator=, 116
- Ipopt::CachedResults< T >, 112
- Ipopt::CompoundMatrix, 151
 - ~CompoundMatrix, 153
 - AddMSInvZImpl, 154
 - Comp, 155
 - CompoundMatrix, 153
 - comps_, 155
 - ComputeColAMaxImpl, 155
 - ComputeRowAMaxImpl, 154
 - const_comps_, 155
 - ConstComp, 155
 - CreateBlockFromSpace, 153
 - GetComp, 153
 - GetCompNonConst, 153
 - HasValidNumbersImpl, 154
 - matrices_valid_, 155
 - MatricesValid, 155
 - MultVectorImpl, 154

- NComps_Cols, 154
- NComps_Rows, 154
- operator=, 155
- owner_space_, 155
- PrintImpl, 155
- SetComp, 153
- SetCompNonConst, 153
- SinvBlrmZMTdBrlImpl, 154
- TransMultVectorImpl, 154
- Ipopt::CompoundMatrixSpace, 156
 - ~CompoundMatrixSpace, 158
 - allocate_block_, 160
 - block_cols_, 160
 - block_rows_, 160
 - comp_spaces_, 160
 - CompoundMatrixSpace, 158
 - Diagonal, 159
 - diagonal_, 160
 - dimensions_set_, 159
 - DimensionsSet, 159
 - GetBlockCols, 158
 - GetBlockRows, 158
 - GetCompSpace, 158
 - MakeNew, 159
 - MakeNewCompoundMatrix, 159
 - NComps_Cols, 159
 - NComps_Rows, 159
 - ncomps_cols_, 159
 - ncomps_rows_, 159
 - operator=, 159
 - SetBlockCols, 158
 - SetBlockRows, 158
 - SetCompSpace, 158
- Ipopt::CompoundSymMatrix, 160
 - ~CompoundSymMatrix, 162
 - Comp, 164
 - CompoundSymMatrix, 162, 163
 - comps_, 164
 - ComputeRowAMaxImpl, 164
 - const_comps_, 164
 - ConstComp, 164
 - GetComp, 163
 - GetCompNonConst, 163
 - IsValidNumbersImpl, 164
 - MakeNewCompoundSymMatrix, 163
 - matrices_valid_, 165
 - MatricesValid, 164
 - MultVectorImpl, 163
 - NComps_Dim, 163
 - operator=, 164
 - owner_space_, 164
 - PrintImpl, 164
 - SetComp, 163
 - SetCompNonConst, 163
- Ipopt::CompoundSymMatrixSpace, 165
 - ~CompoundSymMatrixSpace, 167
 - allocate_block_, 168
 - block_dim_, 168
 - comp_spaces_, 168
 - CompoundSymMatrixSpace, 167
 - dimensions_set_, 168
 - DimensionsSet, 168
 - GetBlockDim, 167
 - GetCompSpace, 167
 - MakeNewCompoundSymMatrix, 167
 - MakeNewSymMatrix, 167
 - NComps_Dim, 167
 - ncomp_spaces_, 168
 - operator=, 168
 - SetBlockDim, 167
 - SetCompSpace, 167
- Ipopt::CompoundVector, 168
 - ~CompoundVector, 171
 - AddScalarImpl, 174
 - AddTwoVectorsImpl, 174
 - AddVectorQuotientImpl, 175
 - AmaxImpl, 173
 - AsumImpl, 173
 - AxpyImpl, 173
 - Comp, 175
 - CompoundVector, 171, 172
 - comps_, 175
 - const_comps_, 175
 - ConstComp, 175
 - CopyImpl, 172
 - DotImpl, 173
 - ElementWiseAbsImpl, 174
 - ElementWiseDivideImpl, 173
 - ElementWiseMaxImpl, 173
 - ElementWiseMinImpl, 173
 - ElementWiseMultiplyImpl, 173
 - ElementWiseReciprocalImpl, 174
 - ElementWiseSgnImpl, 174
 - ElementWiseSqrtImpl, 174
 - FracToBoundImpl, 175
 - GetComp, 172
 - GetCompNonConst, 172
 - IsValidNumbersImpl, 175
 - IsCompConst, 172
 - IsCompNull, 172
 - MaxImpl, 174
 - MinImpl, 174
 - NComps, 172
 - Nrm2Impl, 173
 - operator=, 175
 - owner_space_, 176
 - PrintImpl, 175
 - ScallImpl, 172

- SetComp, 172
- SetCompNonConst, 172
- SetImpl, 173
- SumImpl, 174
- SumLogsImpl, 174
- vectors_valid_, 176
- VectorsValid, 175
- Ipopt::CompoundVectorSpace, 176
 - ~CompoundVectorSpace, 177
 - comp_spaces_, 179
 - CompoundVectorSpace, 177
 - GetCompSpace, 179
 - MakeNew, 179
 - MakeNewCompoundVector, 179
 - NCompSpaces, 179
 - ncomp_spaces_, 179
 - operator=, 179
 - SetCompSpace, 178
- Ipopt::ConvergenceCheck, 180
 - ~ConvergenceCheck, 181
 - CheckConvergence, 182
 - ConvergenceCheck, 181
 - ConvergenceStatus, 181
 - CurrentIsAcceptable, 182
 - InitializeImpl, 182
 - operator=, 182
- Ipopt::DefaultIterateInitializer, 182
 - ~DefaultIterateInitializer, 185
 - aug_system_solver_, 187
 - bound_frac_, 186
 - bound_mult_init_method_, 187
 - bound_mult_init_val_, 187
 - bound_push_, 186
 - BoundMultInitMethod, 185
 - CalculateLeastSquareDuals, 186
 - CalculateLeastSquarePrimals, 186
 - constr_mult_init_max_, 186
 - DefaultIterateInitializer, 185
 - eq_mult_calculator_, 187
 - InitializeImpl, 185
 - least_square_init_duals_, 187
 - least_square_init_primal_, 187
 - least_square_mults, 186
 - mu_init_, 187
 - operator=, 186
 - push_variables, 185
 - RegisterOptions, 186
 - SetInitialIterates, 185
 - slack_bound_frac_, 186
 - slack_bound_push_, 186
 - warm_start_init_point_, 187
 - warm_start_initializer_, 187
- Ipopt::DenseGenMatrix, 188
 - ~DenseGenMatrix, 190
- AddMatrixProduct, 191
- CholeskyBackSolveMatrix, 192
- CholeskySolveMatrix, 192
- CholeskySolveVector, 192
- ComputeCholeskyFactor, 191
- ComputeColAMaxImpl, 193
- ComputeEigenVectors, 192
- ComputeLUFactorInPlace, 192
- ComputeRowAMaxImpl, 193
- Copy, 191
- DenseGenMatrix, 190, 191
- Factorization, 190
- factorization_, 193
- FillIdentity, 191
- HasValidNumbersImpl, 193
- HighRankUpdateTranspose, 191
- initialized_, 193
- LUSolveMatrix, 192
- LUSolveVector, 192
- MakeNewDenseGenMatrix, 191
- MultVectorImpl, 192
- operator=, 193
- owner_space_, 193
- pivot_, 194
- PrintImpl, 193
- ScaleColumns, 191
- TransMultVectorImpl, 192
- Values, 191
- values_, 193
- Ipopt::DenseGenMatrixSpace, 194
 - ~DenseGenMatrixSpace, 195
 - DenseGenMatrixSpace, 195
 - MakeNew, 195
 - MakeNewDenseGenMatrix, 195
- Ipopt::DenseSymMatrix, 195
 - ~DenseSymMatrix, 197
 - AddMatrix, 198
 - ComputeRowAMaxImpl, 198
 - DenseSymMatrix, 197
 - FillIdentity, 198
 - HasValidNumbersImpl, 198
 - HighRankUpdate, 198
 - HighRankUpdateTranspose, 198
 - initialized_, 199
 - MakeNewDenseSymMatrix, 197
 - MultVectorImpl, 198
 - operator=, 199
 - owner_space_, 199
 - PrintImpl, 198
 - SpecialAddForLMSR1, 198
 - Values, 197
 - values_, 199
- Ipopt::DenseSymMatrixSpace, 199
 - ~DenseSymMatrixSpace, 200

- DenseSymMatrixSpace, 200
- MakeNewDenseSymMatrix, 200
- MakeNewSymMatrix, 200
- Ipopt::DenseVector, 200
 - ~DenseVector, 204
 - AddScalarImpl, 207
 - AddTwoVectorsImpl, 207
 - AddVectorQuotientImpl, 208
 - AmaxImpl, 206
 - AsumImpl, 206
 - AxpyImpl, 205
 - CopyFromPos, 205
 - CopyImpl, 205
 - CopyToPos, 205
 - DenseVector, 204
 - DotImpl, 205
 - ElementWiseAbsImpl, 206
 - ElementWiseDivideImpl, 206
 - ElementWiseMaxImpl, 206
 - ElementWiseMinImpl, 206
 - ElementWiseMultiplyImpl, 206
 - ElementWiseReciprocalImpl, 206
 - ElementWiseSgnImpl, 207
 - ElementWiseSqrtImpl, 207
 - expanded_values_, 208
 - ExpandedValues, 205
 - FracToBoundImpl, 207
 - homogeneous_, 209
 - initialized_, 209
 - IsHomogeneous, 205
 - MakeNewDenseVector, 204
 - MaxImpl, 207
 - MinImpl, 207
 - Nrm2Impl, 206
 - operator=, 208
 - owner_space_, 208
 - ParVector, 208
 - PrintImpl, 208
 - PrintImplOffset, 208
 - ScallImpl, 205
 - Scalar, 205
 - scalar_, 209
 - set_values_from_scalar, 208
 - SetImpl, 206
 - SetValues, 204
 - SumImpl, 207
 - SumLogsImpl, 207
 - Values, 204
 - values_, 208
 - values_allocated, 208
- Ipopt::DenseVectorSpace, 209
 - ~DenseVectorSpace, 211
 - AllocateInternalStorage, 211
 - DenseVectorSpace, 211
 - FreeInternalStorage, 211
 - GetIntegerMetaData, 212
 - GetNumericMetaData, 212
 - GetStringMetaData, 211, 212
 - HasIntegerMetaData, 211
 - HasNumericMetaData, 211
 - HasStringMetaData, 211
 - integer_meta_data_, 213
 - MakeNew, 211
 - MakeNewDenseVector, 211
 - numeric_meta_data_, 213
 - SetIntegerMetaData, 212
 - SetNumericMetaData, 212
 - SetStringMetaData, 212
 - string_meta_data_, 212
- Ipopt::DependentResult
 - ~DependentResult, 214
 - DebugPrint, 215
 - dependent_tags_, 216
 - DependentResult, 214, 215
 - DependentsIdentical, 215
 - GetResult, 215
 - Invalidate, 215
 - IsStale, 215
 - operator=, 215
 - RecieveNotification, 215
 - result_, 216
 - scalar_dependents_, 216
 - stale_, 216
- Ipopt::DependentResult< T >, 213
- Ipopt::DiagMatrix, 216
 - ~DiagMatrix, 218
 - ComputeRowAMaxImpl, 219
 - diag_, 219
 - DiagMatrix, 218
 - GetDiag, 218
 - HasValidNumbersImpl, 219
 - MultVectorImpl, 218
 - operator=, 219
 - PrintImpl, 219
 - SetDiag, 218
- Ipopt::DiagMatrixSpace, 219
 - ~DiagMatrixSpace, 220
 - DiagMatrixSpace, 220, 221
 - MakeNewDiagMatrix, 221
 - MakeNewSymMatrix, 221
 - operator=, 221
- Ipopt::EqMultiplierCalculator, 221
 - ~EqMultiplierCalculator, 222
 - CalculateMultipliers, 223
 - EqMultiplierCalculator, 222
 - InitializeImpl, 223
 - operator=, 223
- Ipopt::EquilibrationScaling, 223

- ~EquilibrationScaling, 224
- DetermineScalingParametersImpl, 225
- EquilibrationScaling, 224, 225
- InitializeImpl, 225
- nlp_, 225
- operator=, 225
- point_perturbation_radius_, 225
- RegisterOptions, 225
- Ipopt::ExactHessianUpdater, 226
 - ~ExactHessianUpdater, 227
 - ExactHessianUpdater, 227
 - InitializeImpl, 227
 - operator=, 227
 - UpdateHessian, 227
- Ipopt::ExpandedMultiVectorMatrix, 227
 - ~ExpandedMultiVectorMatrix, 229
 - ComputeColAMaxImpl, 231
 - ComputeRowAMaxImpl, 230
 - ExpandedMultiVectorMatrix, 229
 - ExpandedMultiVectorMatrixOwnerSpace, 230
 - GetExpansionMatrix, 230
 - GetVector, 230
 - IsValidNumbersImpl, 230
 - MakeNewExpandedMultiVectorMatrix, 230
 - MultVectorImpl, 230
 - operator=, 231
 - owner_space_, 231
 - PrintImpl, 231
 - RowVectorSpace, 230
 - SetVector, 230
 - TransMultVectorImpl, 230
 - vecs_, 231
- Ipopt::ExpandedMultiVectorMatrixSpace, 231
 - ~ExpandedMultiVectorMatrixSpace, 232
 - exp_matrix_, 233
 - ExpandedMultiVectorMatrixSpace, 232
 - GetExpansionMatrix, 233
 - MakeNew, 233
 - MakeNewExpandedMultiVectorMatrix, 233
 - RowVectorSpace, 233
 - vec_space_, 233
- Ipopt::ExpansionMatrix, 233
 - ~ExpansionMatrix, 235
 - AddMSinvZImpl, 236
 - CompressedPosIndices, 236
 - ComputeColAMaxImpl, 237
 - ComputeRowAMaxImpl, 236
 - ExpandedPosIndices, 236
 - ExpansionMatrix, 235
 - MultVectorImpl, 236
 - operator=, 237
 - owner_space_, 237
 - ParExpansionMatrix, 237
 - PrintImpl, 237
 - PrintImplOffset, 237
 - SinvBlrmZMTdBImpl, 236
 - TransMultVectorImpl, 236
- Ipopt::ExpansionMatrixSpace, 237
 - ~ExpansionMatrixSpace, 238
 - compressed_pos_, 239
 - CompressedPosIndices, 239
 - expanded_pos_, 239
 - ExpandedPosIndices, 239
 - ExpansionMatrixSpace, 238
 - MakeNew, 239
 - MakeNewExpansionMatrix, 239
- Ipopt::FileJournal, 240
 - ~FileJournal, 241
 - file_, 242
 - FileJournal, 241
 - FlushBufferImpl, 241
 - Open, 241
 - operator=, 242
 - PrintImpl, 241
 - PrintfImpl, 241
- Ipopt::Filter, 242
 - ~Filter, 243
 - Acceptable, 243, 244
 - AddEntry, 244
 - Clear, 244
 - dim_, 244
 - Filter, 243
 - filter_list_, 244
 - operator=, 244
 - Print, 244
- Ipopt::FilterEntry, 244
 - ~FilterEntry, 245
 - Acceptable, 246
 - Dominated, 246
 - FilterEntry, 245, 246
 - iter, 246
 - iter_, 246
 - operator=, 246
 - val, 246
 - vals_, 246
- Ipopt::FilterLSAcceptor, 247
 - ~FilterLSAcceptor, 250
 - alpha_min_frac_, 254
 - ArmijoHolds, 252
 - AugmentFilter, 253
 - CalculateAlphaMin, 251
 - CheckAcceptabilityOfTrialPoint, 251
 - corrector_compl_avg_red_fact_, 254
 - corrector_type_, 254
 - CorrectorTypeEnum, 250
 - count_successive_filter_rejections_, 255
 - delta_, 253
 - eta_phi_, 253

- filter_, 255
- filter_reset_trigger_, 255
- FilterLSAcceptor, 250
- gamma_phi_, 253
- gamma_theta_, 254
- InitThisLineSearch, 251
- InitialzeImpl, 251
- IsAcceptableToCurrentFilter, 252
- IsAcceptableToCurrentIterate, 252
- IsFtype, 252
- kappa_soc_, 254
- last_rejection_due_to_filter_, 255
- max_filter_resets_, 254
- max_soc_, 254
- n_filter_resets_, 256
- obj_max_inc_, 254
- operator=, 252
- pd_solver_, 256
- PrepareRestoPhaseStart, 251
- reference_barr_, 255
- reference_gradBarrTDelta_, 255
- reference_theta_, 255
- RegisterOptions, 252
- Reset, 251
- s_phi_, 253
- s_theta_, 253
- skip_corr_if_neg_curv_, 254
- skip_corr_in_monotone_mode_, 254
- StartWatchDog, 252
- StopWatchDog, 252
- theta_max_, 253
- theta_max_fact_, 253
- theta_min_, 253
- theta_min_fact_, 253
- TryCorrector, 252
- TrySecondOrderCorrection, 251
- UpdateForNextIteration, 252
- watchdog_barr_, 255
- watchdog_gradBarrTDelta_, 255
- watchdog_theta_, 255
- Ipopt::GenAugSystemSolver, 256
 - ~GenAugSystemSolver, 258
 - AugmentedSystemChanged, 259
 - d_c_tag_, 260
 - d_d_tag_, 261
 - d_s_tag_, 260
 - d_x_tag_, 260
 - dc_vals_copy_, 261
 - dd_vals_copy_, 261
 - delta_c_, 261
 - delta_d_, 261
 - delta_s_, 260
 - delta_x_, 260
 - ds_vals_copy_, 261
 - dx_vals_copy_, 261
 - GenAugSystemSolver, 258
 - IncreaseQuality, 259
 - InitialzeImpl, 259
 - j_c_tag_, 260
 - j_d_tag_, 261
 - MultiSolve, 259
 - NumberOfNegEvals, 259
 - operator=, 259
 - ProvidesInertia, 259
 - solver_interface_, 260
 - UpdateTags, 259
 - w_factor_, 260
 - w_tag_, 260
 - warm_start_same_structure_, 261
- Ipopt::GenKKTSolverInterface, 262
 - ~GenKKTSolverInterface, 262
 - GenKKTSolverInterface, 262
 - IncreaseQuality, 264
 - InitialzeImpl, 263
 - MultiSolve, 263
 - NumberOfNegEvals, 264
 - ProvidesInertia, 264
- Ipopt::GenTMatrix, 264
 - ~GenTMatrix, 266
 - ComputeColAMaxImpl, 268
 - ComputeRowAMaxImpl, 268
 - GenTMatrix, 266, 267
 - HasValidNumbersImpl, 268
 - initialized_, 269
 - lrows, 267
 - jcols, 267
 - MultVectorImpl, 267
 - Nonzeros, 267
 - operator=, 268
 - owner_space_, 268
 - ParGenMatrix, 268
 - PrintImpl, 268
 - PrintImplOffset, 268
 - SetValues, 267
 - TransMultVectorImpl, 267
 - Values, 267
 - values_, 269
- Ipopt::GenTMatrixSpace, 269
 - ~GenTMatrixSpace, 270
 - AllocateInternalStorage, 271
 - FreeInternalStorage, 271
 - GenTMatrix, 271
 - GenTMatrixSpace, 270
 - iRows_, 271
 - lrows, 271
 - jCols_, 271
 - jcols, 271
 - MakeNew, 271

- MakeNewGenTMatrix, [271](#)
- nonZeros_, [271](#)
- Nonzeros, [271](#)
- Ipopt::GradientScaling, [272](#)
 - ~GradientScaling, [273](#)
 - DetermineScalingParametersImpl, [274](#)
 - GradientScaling, [273](#)
 - InitializImpl, [274](#)
 - nlp_, [274](#)
 - operator=, [274](#)
 - RegisterOptions, [274](#)
 - scaling_constr_target_gradient_, [274](#)
 - scaling_max_gradient_, [274](#)
 - scaling_min_value_, [274](#)
 - scaling_obj_target_gradient_, [274](#)
- Ipopt::HessianUpdater, [275](#)
 - ~HessianUpdater, [276](#)
 - HessianUpdater, [276](#)
 - InitializImpl, [276](#)
 - operator=, [276](#)
 - UpdateHessian, [276](#)
- Ipopt::IdentityMatrix, [276](#)
 - ~IdentityMatrix, [278](#)
 - AddMSInvZImpl, [279](#)
 - ComputeRowAMaxImpl, [279](#)
 - Dim, [279](#)
 - factor_, [279](#)
 - GetFactor, [278](#)
 - HasValidNumbersImpl, [279](#)
 - IdentityMatrix, [278](#)
 - MultVectorImpl, [279](#)
 - operator=, [279](#)
 - PrintImpl, [279](#)
 - SetFactor, [278](#)
- Ipopt::IdentityMatrixSpace, [280](#)
 - ~IdentityMatrixSpace, [281](#)
 - IdentityMatrixSpace, [281](#)
 - MakeNewIdentityMatrix, [281](#)
 - MakeNewSymMatrix, [281](#)
 - operator=, [281](#)
- Ipopt::InexactAlgorithmBuilder, [281](#)
 - ~InexactAlgorithmBuilder, [283](#)
 - BuildBasicAlgorithm, [283](#)
 - BuildIpoptObjects, [283](#)
 - custom_solver_, [283](#)
 - InexactAlgorithmBuilder, [283](#)
 - operator=, [283](#)
 - RegisterOptions, [283](#)
- Ipopt::InexactCq, [284](#)
 - ~InexactCq, [286](#)
 - curr_W_times_vec_s, [287](#)
 - curr_W_times_vec_s_cache_, [288](#)
 - curr_W_times_vec_x, [287](#)
 - curr_W_times_vec_x_cache_, [288](#)
 - curr_Wu_s, [287](#)
 - curr_Wu_s_cache_, [288](#)
 - curr_Wu_x, [287](#)
 - curr_Wu_x_cache_, [288](#)
 - curr_jac_cdT_times_curr_cdminuss, [286](#)
 - curr_jac_cdT_times_curr_cdminuss_cache_, [288](#)
 - curr_jac_times_normal_c, [287](#)
 - curr_jac_times_normal_c_cache_, [289](#)
 - curr_jac_times_normal_d, [287](#)
 - curr_jac_times_normal_d_cache_, [289](#)
 - curr_scaled_A_norm2, [287](#)
 - curr_scaled_Ac_norm, [287](#)
 - curr_scaled_Ac_norm_cache_, [288](#)
 - curr_scaling_slacks, [287](#)
 - curr_scaling_slacks_cache_, [288](#)
 - curr_slack_scaled_d_minus_s, [287](#)
 - curr_slack_scaled_d_minus_s_cache_, [288](#)
 - curr_uWu, [287](#)
 - curr_uWu_cache_, [289](#)
 - InexData, [288](#)
 - InexactCq, [286](#)
 - Initialize, [286](#)
 - ip_cq_, [288](#)
 - ip_data_, [288](#)
 - ip_nlp_, [288](#)
 - operator=, [287](#)
 - RegisterOptions, [286](#)
 - slack_scale_max_, [289](#)
 - slack_scaled_norm, [287](#)
 - slack_scaled_norm_cache_, [288](#)
- Ipopt::InexactData, [289](#)
 - ~InexactData, [291](#)
 - AcceptTrialPoint, [292](#)
 - compute_normal, [293](#)
 - compute_normal_, [293](#)
 - curr_nu, [292](#)
 - curr_nu_, [293](#)
 - full_step_accepted, [292](#)
 - full_step_accepted_, [293](#)
 - InexactData, [291](#)
 - Initialize, [291](#)
 - InitializeDataStructures, [291](#)
 - next_compute_normal, [293](#)
 - next_compute_normal_, [294](#)
 - normal_s, [292](#)
 - normal_s_, [293](#)
 - normal_x, [292](#)
 - normal_x_, [293](#)
 - operator=, [293](#)
 - set_compute_normal, [293](#)
 - set_curr_nu, [292](#)
 - set_full_step_accepted, [292](#)
 - set_next_compute_normal, [293](#)
 - set_normal_s, [292](#)

- set_normal_x, 292
- set_tangential_s, 292
- set_tangential_x, 292
- tangential_s, 292
- tangential_s_, 293
- tangential_x, 292
- tangential_x_, 293
- Ipopt::InexactDoglegNormalStep, 294
 - ~InexactDoglegNormalStep, 295
 - ComputeNormalStep, 296
 - curr_omega_, 296
 - InexactDoglegNormalStep, 295
 - InitialzeImpl, 296
 - last_tr_inactive_, 296
 - newton_step_, 296
 - normal_tester_, 296
 - omega_max_, 296
 - operator=, 296
 - RegisterOptions, 296
- Ipopt::InexactLSAcceptor, 297
 - ~InexactLSAcceptor, 300
 - accepted_by_low_only_, 305
 - CalcPred, 302
 - CalculateAlphaMin, 301
 - CheckAcceptabilityOfTrialPoint, 301
 - ComputeAlphaForY, 302
 - eta_, 303
 - flexible_penalty_function_, 303
 - HasComputeAlphaForY, 302
 - in_tt2_, 305
 - InexCq, 302
 - InexData, 302
 - inexact_decomposition_activate_tol_, 304
 - inexact_decomposition_inactivate_tol_, 304
 - InexactLSAcceptor, 300
 - InitThisLineSearch, 300
 - InitialzeImpl, 300
 - IsAcceptableToCurrentIterate, 302
 - last_nu_, 304
 - last_nu_low_, 304
 - nu_, 304
 - nu_inc_, 303
 - nu_init_, 302
 - nu_low_, 304
 - nu_low_fact_, 303
 - nu_low_init_, 303
 - nu_update_inf_skip_tol_, 303
 - operator=, 302
 - PrepareRestoPhaseStart, 301
 - reference_barr_, 303
 - reference_pred_, 304
 - reference_theta_, 303
 - RegisterOptions, 302
 - Reset, 300
 - ResetSlacks, 302
 - resto_pred_, 305
 - rho_, 303
 - StartWatchDog, 301
 - StopWatchDog, 301
 - tcc_theta_, 303
 - TryCorrector, 301
 - TrySecondOrderCorrection, 301
 - UpdateForNextIteration, 301
 - watchdog_barr_, 304
 - watchdog_pred_, 304
 - watchdog_theta_, 304
- Ipopt::InexactNewtonNormalStep, 305
 - ~InexactNewtonNormalStep, 306
 - aug_solver_, 307
 - ComputeNewtonNormalStep, 307
 - InexCq, 307
 - InexData, 307
 - InexactNewtonNormalStep, 306, 307
 - InitialzeImpl, 307
 - operator=, 307
 - RegisterOptions, 307
- Ipopt::InexactNormalStepCalculator, 308
 - ~InexactNormalStepCalculator, 309
 - ComputeNormalStep, 309
 - InexCq, 309
 - InexData, 309
 - InexactNormalStepCalculator, 309
 - InitialzeImpl, 309
 - operator=, 309
- Ipopt::InexactNormalTerminationTester, 310
 - ~InexactNormalTerminationTester, 311
 - c_Avc_norm_cauchy_, 313
 - Clear, 312
 - GetSolverIterations, 312
 - inexact_normal_max_iter_, 313
 - inexact_normal_tol_, 313
 - InexactNormalTerminationTester, 311
 - InitialzeImpl, 312
 - InitializeSolve, 312
 - last_iter_, 313
 - operator=, 312
 - RegisterOptions, 312
 - requires_scaling_, 313
 - Set_c_Avc_norm_cauchy, 312
 - TestTermination, 312
- Ipopt::InexactPDSolver, 313
 - ~InexactPDSolver, 315
 - augSysSolver_, 316
 - ComputeResiduals, 316
 - HessianRequiresChange, 316
 - InexCq, 316
 - InexData, 316
 - inexact_regularization_ls_count_trigger_, 317

- InexactPDSolver, 315
- InitializeImpl, 315
- is_pardiso_, 317
- last_info_ls_count_, 317
- modify_hessian_with_slacks_, 316
- operator=, 315
- perturbHandler_, 316
- RegisterOptions, 315
- Solve, 315
- tcc_psi_, 316
- tcc_theta_, 316
- tcc_theta_mu_exponent_, 316
- Ipopt::InexactPDTerminationTester, 317
 - ~InexactPDTerminationTester, 319
 - c_norm_, 322
 - c_plus_Av_norm_, 322
 - Clear, 320
 - curr_Av_c_, 322
 - curr_Av_d_, 322
 - curr_Av_norm_, 322
 - curr_Wv_s_, 323
 - curr_Wv_x_, 323
 - curr_grad_barrier_obj_s_, 322
 - curr_grad_barrier_obj_x_, 322
 - curr_jac_c_, 322
 - curr_jac_d_, 322
 - curr_scaling_slacks_, 322
 - curr_tt1_norm_, 322
 - curr_tt2_norm_, 323
 - GetSolverIterations, 320
 - inexact_desired_pd_residual_, 321
 - inexact_desired_pd_residual_iter_, 321
 - InexactPDTerminationTester, 319
 - InitializeImpl, 320
 - InitializeSolve, 320
 - last_Av_norm_, 323
 - last_iter_, 323
 - last_tt1_norm_, 323
 - operator=, 320
 - RegisterOptions, 320
 - requires_scaling_, 321
 - rho_, 321
 - tcc_psi_, 320
 - tcc_theta_, 320
 - tcc_theta_mu_exponent_, 321
 - tcc_zeta_, 321
 - TestTermination, 320
 - try_tt2_, 323
 - tt_eps2_, 321
 - tt_eps3_, 321
 - tt_kappa1_, 321
 - tt_kappa2_, 321
 - v_norm_scaled_, 322
- Ipopt::InexactSearchDirCalculator, 323
 - ~InexactSearchDirCalculator, 325
 - ComputeSearchDirection, 326
 - decomposition_type_, 327
 - DecompositionTypeEnum, 325
 - InexCq, 326
 - InexData, 326
 - inexact_pd_solver_, 326
 - InexactSearchDirCalculator, 325
 - InitializeImpl, 326
 - local_inf_Ac_tol_, 326
 - normal_step_calculator_, 326
 - operator=, 326
 - RegisterOptions, 326
- Ipopt::InexactTSymScalingMethod, 327
 - ~InexactTSymScalingMethod, 328
 - ComputeSymTScalingFactors, 328
 - InexCq, 328
 - InexactTSymScalingMethod, 328
 - InitializeImpl, 328
 - operator=, 328
- Ipopt::IpoptAdditionalCq, 329
 - ~IpoptAdditionalCq, 330
 - Initialize, 330
 - IpoptAdditionalCq, 330
 - operator=, 330
- Ipopt::IpoptAdditionalData, 330
 - ~IpoptAdditionalData, 331
 - AcceptTrialPoint, 332
 - Initialize, 332
 - InitializeDataStructures, 332
 - IpoptAdditionalData, 331
 - operator=, 332
- Ipopt::IpoptAlgorithm, 332
 - ~IpoptAlgorithm, 335
 - AcceptTrialPoint, 336
 - calc_number_of_bounds, 336
 - ComputeAcceptableTrialPoint, 336
 - ComputeFeasibilityMultipliers, 336
 - ComputeSearchDirection, 336
 - conv_check_, 337
 - correct_bound_multiplier, 336
 - eq_multiplier_calculator_, 337
 - hessian_updater_, 337
 - InitializeImpl, 335
 - InitializeIterates, 336
 - IpoptAlgorithm, 335
 - iter_output_, 337
 - iterate_initializer_, 337
 - kappa_sigma_, 337
 - line_search_, 337
 - linear_solver_, 338
 - mehrotra_algorithm_, 338
 - mu_update_, 337
 - operator=, 336

- Optimize, [335](#)
- OutputIteration, [336](#)
- print_copyright_message, [336](#)
- PrintProblemStatistics, [336](#)
- recalc_y_, [337](#)
- recalc_y_feas_tol_, [338](#)
- RegisterOptions, [335](#)
- search_dir_calculator_, [337](#)
- SearchDirCalc, [335](#)
- skip_print_problem_stats_, [337](#)
- UpdateBarrierParameter, [336](#)
- UpdateHessian, [336](#)
- Ipopt::IpoptApplication, [338](#)
 - ~IpoptApplication, [341](#)
 - alg_, [344](#)
 - AlgorithmObject, [343](#)
 - call_optimize, [344](#)
 - clone, [341](#)
 - inexact_algorithm_, [345](#)
 - Initialize, [341](#), [342](#)
 - ip_cq_, [345](#)
 - ip_data_, [344](#)
 - ip_nlp_, [344](#)
 - IpoptApplication, [341](#)
 - IpoptCQObject, [343](#)
 - IpoptDataObject, [343](#)
 - IpoptNLPObject, [343](#)
 - Jnlst, [342](#)
 - jnlst_, [344](#)
 - nlp_adapter_, [345](#)
 - OpenOutputFile, [342](#)
 - operator=, [343](#)
 - OptimizeNLP, [342](#)
 - OptimizeTNLP, [342](#)
 - Options, [343](#)
 - options_, [344](#)
 - PrintCopyrightMessage, [343](#)
 - ReOptimizeNLP, [342](#)
 - ReOptimizeTNLP, [342](#)
 - read_params_dat_, [344](#)
 - reg_options_, [344](#)
 - RegOptions, [342](#)
 - RegisterAllIpoptOptions, [343](#)
 - RegisterOptions, [343](#)
 - replace_bounds_, [345](#)
 - rethrow_nonipoptexception_, [344](#)
 - RethrowNonIpoptException, [343](#)
 - Statistics, [343](#)
 - statistics_, [344](#)
- Ipopt::IpoptCalculatedQuantities, [345](#)
 - ~IpoptCalculatedQuantities, [355](#)
 - add_cq_, [365](#)
 - AdditionalCq, [364](#)
 - AdjustedTrialSlacks, [356](#)
 - CalcBarrierTerm, [364](#)
 - CalcCentralityMeasure, [362](#)
 - CalcCompl, [364](#)
 - CalcFracToBound, [364](#)
 - CalcNormOfType, [363](#)
 - CalcSlack_L, [364](#)
 - CalcSlack_U, [364](#)
 - CalculateSafeSlack, [365](#)
 - ComputeDampingIndicators, [365](#)
 - ComputeOptimalityErrorScaling, [365](#)
 - constr_viol_normtype, [363](#)
 - constr_viol_normtype_, [366](#)
 - curr_avg_compl, [363](#)
 - curr_avg_compl_cache_, [372](#)
 - curr_barrier_error, [362](#)
 - curr_barrier_error_cache_, [372](#)
 - curr_barrier_obj, [357](#)
 - curr_barrier_obj_cache_, [367](#)
 - curr_c, [357](#)
 - curr_c_cache_, [368](#)
 - curr_centrality_measure, [362](#)
 - curr_centrality_measure_cache_, [372](#)
 - curr_compl_s_L, [360](#)
 - curr_compl_s_L_cache_, [370](#)
 - curr_compl_s_U, [360](#)
 - curr_compl_s_U_cache_, [370](#)
 - curr_compl_x_L, [360](#)
 - curr_compl_x_L_cache_, [370](#)
 - curr_compl_x_U, [360](#)
 - curr_compl_x_U_cache_, [370](#)
 - curr_complementarity, [361](#)
 - curr_complementarity_cache_, [372](#)
 - curr_constraint_violation, [359](#)
 - curr_constraint_violation_cache_, [369](#)
 - curr_d, [358](#)
 - curr_d_cache_, [368](#)
 - curr_d_minus_s, [358](#)
 - curr_d_minus_s_cache_, [368](#)
 - curr_dual_frac_to_the_bound, [363](#)
 - curr_dual_infeasibility, [361](#)
 - curr_dual_infeasibility_cache_, [371](#)
 - curr_exact_hessian, [359](#)
 - curr_exact_hessian_cache_, [369](#)
 - curr_f, [356](#)
 - curr_f_cache_, [367](#)
 - curr_grad_barrier_obj_s, [357](#)
 - curr_grad_barrier_obj_s_cache_, [367](#)
 - curr_grad_barrier_obj_x, [357](#)
 - curr_grad_barrier_obj_x_cache_, [367](#)
 - curr_grad_f, [357](#)
 - curr_grad_f_cache_, [367](#)
 - curr_grad_lag_s, [360](#)
 - curr_grad_lag_s_cache_, [370](#)
 - curr_grad_lag_with_damping_s, [360](#)

curr_grad_lag_with_damping_s_cache_, 370
 curr_grad_lag_with_damping_x, 360
 curr_grad_lag_with_damping_x_cache_, 370
 curr_grad_lag_x, 360
 curr_grad_lag_x_cache_, 370
 curr_gradBarrTDelta, 363
 curr_gradBarrTDelta_cache_, 373
 curr_jac_c, 358
 curr_jac_c_cache_, 368
 curr_jac_c_times_vec, 359
 curr_jac_c_times_vec_cache_, 369
 curr_jac_cT_times_curr_y_c, 358
 curr_jac_cT_times_vec, 358
 curr_jac_cT_times_vec_cache_, 369
 curr_jac_d, 358
 curr_jac_d_cache_, 368
 curr_jac_d_times_vec, 359
 curr_jac_d_times_vec_cache_, 369
 curr_jac_dT_times_curr_y_d, 359
 curr_jac_dT_times_vec, 358
 curr_jac_dT_times_vec_cache_, 369
 curr_nlp_constraint_violation, 359
 curr_nlp_constraint_violation_cache_, 369
 curr_nlp_error, 362
 curr_nlp_error_cache_, 372
 curr_primal_dual_system_error, 362
 curr_primal_dual_system_error_cache_, 372
 curr_primal_frac_to_the_bound, 362
 curr_primal_infeasibility, 361
 curr_primal_infeasibility_cache_, 371
 curr_relaxed_compl_s_L, 361
 curr_relaxed_compl_s_L_cache_, 371
 curr_relaxed_compl_s_U, 361
 curr_relaxed_compl_s_U_cache_, 371
 curr_relaxed_compl_x_L, 361
 curr_relaxed_compl_x_L_cache_, 371
 curr_relaxed_compl_x_U, 361
 curr_relaxed_compl_x_U_cache_, 371
 curr_sigma_s, 363
 curr_sigma_s_cache_, 372
 curr_sigma_x, 363
 curr_sigma_x_cache_, 372
 curr_slack_s_L, 356
 curr_slack_s_L_cache_, 366
 curr_slack_s_U, 356
 curr_slack_s_U_cache_, 366
 curr_slack_x_L, 356
 curr_slack_x_L_cache_, 366
 curr_slack_x_U, 356
 curr_slack_x_U_cache_, 366
 dampind_s_L_, 373
 dampind_s_U_, 373
 dampind_x_L_, 373
 dampind_x_U_, 373
 dual_frac_to_the_bound, 362
 dual_frac_to_the_bound_cache_, 372
 GetIpoptNLP, 363
 grad_kappa_times_damping_s, 357
 grad_kappa_times_damping_s_cache_, 368
 grad_kappa_times_damping_x, 357
 grad_kappa_times_damping_x_cache_, 368
 HaveAddCq, 355
 in_restoration_phase, 365
 Initialize, 356
 initialize_called_, 374
 ip_data_, 365
 ip_nlp_, 365
 IpoptCalculatedQuantities, 355
 IsSquareProblem, 363
 kappa_d_, 366
 mu_target_, 366
 num_adjusted_slack_s_L_, 367
 num_adjusted_slack_s_U_, 367
 num_adjusted_slack_x_L_, 367
 num_adjusted_slack_x_U_, 367
 operator=, 364
 primal_frac_to_the_bound, 362
 primal_frac_to_the_bound_cache_, 372
 RegisterOptions, 364
 ResetAdjustedTrialSlacks, 356
 s_max_, 365
 SetAddCq, 355
 slack_move_, 366
 Tmp_c, 364
 tmp_c_, 373
 Tmp_d, 364
 tmp_d_, 373
 Tmp_s, 364
 tmp_s_, 373
 Tmp_s_L, 364
 tmp_s_L_, 374
 Tmp_s_U, 364
 tmp_s_U_, 374
 Tmp_x, 364
 tmp_x_, 373
 Tmp_x_L, 364
 tmp_x_L_, 373
 Tmp_x_U, 364
 tmp_x_U_, 374
 trial_avg_compl, 363
 trial_avg_compl_cache_, 373
 trial_barrier_obj, 357
 trial_barrier_obj_cache_, 367
 trial_c, 357
 trial_c_cache_, 368
 trial_compl_s_L, 360
 trial_compl_s_L_cache_, 371
 trial_compl_s_U, 361

trial_compl_s_U_cache_, 371
 trial_compl_x_L, 360
 trial_compl_x_L_cache_, 370
 trial_compl_x_U, 360
 trial_compl_x_U_cache_, 371
 trial_complementarity, 361
 trial_complementarity_cache_, 372
 trial_constraint_violation, 359
 trial_constraint_violation_cache_, 369
 trial_d, 358
 trial_d_cache_, 368
 trial_d_minus_s, 358
 trial_d_minus_s_cache_, 368
 trial_dual_infeasibility, 361
 trial_dual_infeasibility_cache_, 371
 trial_f, 357
 trial_f_cache_, 367
 trial_grad_f, 357
 trial_grad_f_cache_, 367
 trial_grad_lag_s, 360
 trial_grad_lag_s_cache_, 370
 trial_grad_lag_x, 360
 trial_grad_lag_x_cache_, 370
 trial_jac_c, 358
 trial_jac_c_cache_, 368
 trial_jac_cT_times_trial_y_c, 359
 trial_jac_cT_times_vec, 358
 trial_jac_cT_times_vec_cache_, 369
 trial_jac_d, 358
 trial_jac_d_cache_, 368
 trial_jac_dT_times_trial_y_d, 359
 trial_jac_dT_times_vec, 358
 trial_jac_dT_times_vec_cache_, 369
 trial_primal_dual_system_error, 362
 trial_primal_dual_system_error_cache_, 372
 trial_primal_infeasibility, 361
 trial_primal_infeasibility_cache_, 371
 trial_slack_s_L, 356
 trial_slack_s_L_cache_, 367
 trial_slack_s_U, 356
 trial_slack_s_U_cache_, 367
 trial_slack_x_L, 356
 trial_slack_x_L_cache_, 366
 trial_slack_x_U, 356
 trial_slack_x_U_cache_, 366
 uncached_dual_frac_to_the_bound, 362
 uncached_slack_frac_to_the_bound, 363
 unscaled_curr_c, 357
 unscaled_curr_complementarity, 361
 unscaled_curr_d, 358
 unscaled_curr_dual_infeasibility, 361
 unscaled_curr_dual_infeasibility_cache_, 371
 unscaled_curr_f, 356
 unscaled_curr_nlp_constraint_violation, 359
 unscaled_curr_nlp_constraint_violation_cache_, 369
 unscaled_curr_nlp_error, 362
 unscaled_curr_nlp_error_cache_, 372
 unscaled_trial_c, 357
 unscaled_trial_f, 357
 unscaled_trial_nlp_constraint_violation, 359
 unscaled_trial_nlp_constraint_violation_cache_, 369
 warm_start_same_structure_, 366
 Ipopt::IpoptData, 374
 ~IpoptData, 379
 AcceptTrialPoint, 382
 add_data_, 388
 AdditionalData, 385
 Append_info_string, 384
 CopyTrialToCurrent, 382
 cpu_time_start, 383
 cpu_time_start_, 388
 curr, 380
 curr_, 386
 curr_mu, 382
 curr_mu_, 386
 curr_tau, 382
 curr_tau_, 387
 delta, 380
 delta_, 386
 delta_aff, 381
 delta_aff_, 386
 free_mu_mode_, 387
 FreeMuMode, 383
 getPDPert, 385
 have_affine_deltas_, 386
 have_deltas_, 386
 have_prototypes_, 387
 HaveAddData, 385
 HaveAffineDeltas, 381
 HaveDeltas, 381
 Inc_info_iters_since_header, 384
 info_alpha_dual, 384
 info_alpha_dual_, 388
 info_alpha_primal, 383
 info_alpha_primal_, 387
 info_alpha_primal_char, 383
 info_alpha_primal_char_, 387
 info_iters_since_header, 384
 info_iters_since_header_, 388
 info_last_output, 384
 info_last_output_, 388
 info_ls_count, 384
 info_ls_count_, 388
 info_regu_x, 383
 info_regu_x_, 387
 info_skip_output, 384
 info_skip_output_, 388
 info_string, 384

- info_string_, 388
- Initialize, 380
- initialize_called_, 387
- InitializeDataStructures, 380
- lpoptData, 379
- iter_count, 382
- iter_count_, 386
- iterates_space_, 388
- mu_initialized_, 387
- MuInitialized, 382
- operator=, 385
- pd_pert_c_, 389
- pd_pert_d_, 389
- pd_pert_s_, 389
- pd_pert_x_, 388
- RegisterOptions, 385
- ResetInfo, 385
- Set_W, 381
- set_delta, 380, 381
- set_delta_aff, 381
- Set_info_alpha_dual, 384
- Set_info_alpha_primal, 383
- Set_info_alpha_primal_char, 384
- Set_info_iters_since_header, 385
- Set_info_last_output, 384
- Set_info_ls_count, 384
- Set_info_regu_x, 383
- Set_info_skip_output, 384
- Set_iter_count, 382
- Set_mu, 382
- Set_tau, 382
- Set_tiny_step_flag, 383
- Set_tol, 383
- set_trial, 380
- SetAddData, 385
- SetFreeMuMode, 382
- SetHaveAffineDeltas, 382
- SetHaveDeltas, 381
- setPDPert, 385
- SetTrialBoundMultipliersFromStep, 380
- SetTrialEqMultipliersFromStep, 380
- SetTrialPrimalVariablesFromStep, 380
- tau_initialized_, 387
- TauInitialized, 382
- timing_statistics_, 388
- TimingStats, 385
- tiny_step_flag, 383
- tiny_step_flag_, 387
- tol, 383
- tol_, 387
- trial, 380
- trial_, 386
- W, 381
- W_, 386
- lpopt::lpoptException, 389
 - ~lpoptException, 390
 - file_name_, 391
 - lpoptException, 390
 - line_number_, 391
 - Message, 391
 - msg_, 391
 - operator=, 391
 - ReportException, 391
 - type_, 391
- lpopt::lpoptNLP, 391
 - ~lpoptNLP, 394
 - AdjustVariableBounds, 397
 - c, 395
 - c_evals, 397
 - d, 395
 - d_L, 396
 - d_U, 396
 - d_evals, 397
 - f, 395, 397
 - f_evals, 397
 - FinalizeSolution, 398
 - GetSpaces, 396
 - GetWarmStartIterate, 395
 - grad_f, 395, 398
 - grad_f_evals, 397
 - h, 395, 398
 - h_evals, 397
 - HessianMatrixSpace, 396
 - Initialize, 394
 - InitializeStructures, 395
 - IntermediateCallBack, 398
 - lpoptNLP, 394
 - jac_c, 395
 - jac_c_evals, 397
 - jac_d, 395
 - jac_d_evals, 397
 - NLP_scaling, 398
 - nlp_scaling_, 398
 - objective_depends_on_mu, 397
 - operator=, 398
 - Pd_L, 396
 - Pd_U, 396
 - Px_L, 396
 - Px_U, 396
 - uninitialized_h, 398
 - x_L, 395
 - x_U, 396
- lpopt::IterateInitializer, 399
 - ~IterateInitializer, 400
 - InitializeImpl, 400
 - IterateInitializer, 400
 - operator=, 400
 - SetInitialIterates, 400

- Ipopt::IteratesVector, 400
 - ~IteratesVector, 404
 - create_new_s, 406
 - create_new_s_copy, 406
 - create_new_v_L, 409
 - create_new_v_L_copy, 409
 - create_new_v_U, 410
 - create_new_v_U_copy, 410
 - create_new_x, 405
 - create_new_x_copy, 405
 - create_new_y_c, 406
 - create_new_y_c_copy, 406
 - create_new_y_d, 407
 - create_new_y_d_copy, 407
 - create_new_z_L, 408
 - create_new_z_L_copy, 408
 - create_new_z_U, 408
 - create_new_z_U_copy, 408
 - GetIterateFromComp, 411
 - GetNonConstIterateFromComp, 411
 - GetTagSum, 411
 - IteratesVector, 404
 - MakeNewContainer, 405
 - MakeNewIteratesVector, 405
 - MakeNewIteratesVectorCopy, 405
 - operator=, 411
 - owner_space_, 411
 - s, 406
 - s_NonConst, 406
 - Set_bound_mult, 411
 - Set_bound_mult_NonConst, 411
 - Set_eq_mult, 410
 - Set_eq_mult_NonConst, 410
 - Set_primal, 410
 - Set_primal_NonConst, 410
 - Set_s, 406
 - Set_s_NonConst, 406
 - Set_v_L, 409
 - Set_v_L_NonConst, 409
 - Set_v_U, 410
 - Set_v_U_NonConst, 410
 - Set_x, 405
 - Set_x_NonConst, 405
 - Set_y_c, 407
 - Set_y_c_NonConst, 407
 - Set_y_d, 407
 - Set_y_d_NonConst, 407
 - Set_z_L, 408
 - Set_z_L_NonConst, 408
 - Set_z_U, 409
 - Set_z_U_NonConst, 409
 - v_L, 409
 - v_L_NonConst, 409
 - v_U, 409
 - v_U_NonConst, 410
 - x, 405
 - x_NonConst, 405
 - y_c, 406
 - y_c_NonConst, 406
 - y_d, 407
 - y_d_NonConst, 407
 - z_L, 407
 - z_L_NonConst, 408
 - z_U, 408
 - z_U_NonConst, 408
- Ipopt::IteratesVectorSpace, 412
 - ~IteratesVectorSpace, 413
 - IteratesVectorSpace, 413
 - MakeNew, 414
 - MakeNewCompoundVector, 414
 - MakeNewIteratesVector, 413, 414
 - operator=, 414
 - s_space_, 414
 - SetCompSpace, 414
 - v_L_space_, 415
 - v_U_space_, 415
 - x_space_, 414
 - y_c_space_, 414
 - y_d_space_, 415
 - z_L_space_, 415
 - z_U_space_, 415
- Ipopt::IterationOutput, 415
 - ~IterationOutput, 416
 - InfPrOutput, 416
 - InitializeImpl, 417
 - IterationOutput, 416, 417
 - operator=, 417
 - WriteOutput, 417
- Ipopt::IterativePardisoSolverInterface, 417
 - ~IterativePardisoSolverInterface, 421
 - a_, 423
 - DPARM_, 426
 - debug_cnt_, 427
 - debug_last_iter_, 426
 - decr_factor_, 425
 - dim_, 423
 - Factorization, 423
 - GetValuesArrayPtr, 422
 - have_symbolic_factorization_, 424
 - IPARM_, 426
 - IncreaseQuality, 422
 - InexCq, 423
 - InexData, 423
 - InitializeImpl, 422
 - InitializeStructure, 422
 - initialized_, 426
 - IterativePardisoSolverInterface, 421
 - MAXFCT_, 426

- MNUM_, 426
- MSGVLV_, 426
- MTYPE_, 426
- match_strat_, 424
- MatrixFormat, 422
- MultiSolve, 422
- negevals_, 423
- nonzeros_, 423
- normal_pardiso_iter_coarse_size_, 425
- normal_pardiso_iter_dropping_factor_, 425
- normal_pardiso_iter_dropping_factor_used_, 425
- normal_pardiso_iter_dropping_schur_, 425
- normal_pardiso_iter_dropping_schur_used_, 426
- normal_pardiso_iter_inverse_norm_factor_, 425
- normal_pardiso_iter_max_levels_, 425
- normal_pardiso_iter_max_row_fill_, 425
- normal_pardiso_iter_relative_tol_, 425
- normal_pardiso_max_iter_, 425
- normal_tester_, 427
- NumberOfNegEvals, 422
- operator=, 423
- PT_, 426
- pardiso_iter_coarse_size_, 424
- pardiso_iter_dropping_factor_, 424
- pardiso_iter_dropping_factor_used_, 425
- pardiso_iter_dropping_schur_, 424
- pardiso_iter_dropping_schur_used_, 425
- pardiso_iter_inverse_norm_factor_, 425
- pardiso_iter_max_levels_, 424
- pardiso_iter_max_row_fill_, 425
- pardiso_iter_relative_tol_, 424
- pardiso_max_droptol_corrections_, 424
- pardiso_max_iter_, 424
- pardiso_repeated_perturbation_means_singular_, 424
- PardisoMatchingStrategy, 421
- pd_tester_, 427
- ProvidesInertia, 422
- RegisterOptions, 422
- skip_inertia_check_, 424
- Solve, 423
- SymbolicFactorization, 423
- Ipopt::IterativeSolverTerminationTester, 427
 - ~IterativeSolverTerminationTester, 429
 - Clear, 430
 - ETerminationTest, 429
 - GetJnlst, 430
 - GetSolverIterations, 430
 - GetVectors, 430
 - InexCq, 430
 - InexData, 430
 - InitializeImpl, 429
 - InitializeSolve, 429
 - IterativeSolverTerminationTester, 429
 - operator=, 430
 - TestTermination, 429
- Ipopt::IterativeWsmpSolverInterface, 430
 - ~IterativeWsmpSolverInterface, 433
 - a_, 435
 - DPARAM_, 436
 - dim_, 435
 - Factorization, 434
 - GetValuesArrayPtr, 433
 - have_symbolic_factorization_, 436
 - IPARM_, 436
 - IncreaseQuality, 434
 - InitializeImpl, 433
 - InitializeStructure, 433
 - initialized_, 436
 - InternalSymFact, 434
 - IterativeWsmpSolverInterface, 433
 - matrix_file_number_, 436
 - MatrixFormat, 434
 - MultiSolve, 434
 - NumberOfNegEvals, 434
 - operator=, 434
 - pivtol_changed_, 436
 - ProvidesInertia, 434
 - RegisterOptions, 434
 - Solve, 435
 - SymbolicFactorization, 434
 - wsmp_inexact_droptol_, 435
 - wsmp_inexact_fillin_limit_, 435
 - wsmp_num_threads_, 435
 - wsmp_pivtol_, 435
 - wsmp_pivtolmax_, 435
 - wsmp_scaling_, 435
 - wsmp_write_matrix_iteration_, 435
- Ipopt::Journal, 436
 - ~Journal, 438
 - FlushBuffer, 439
 - FlushBufferImpl, 439
 - IsAccepted, 439
 - Journal, 438
 - Name, 438
 - name_, 439
 - operator=, 439
 - Print, 439
 - print_levels_, 440
 - PrintImpl, 439
 - Printf, 439
 - PrintfImpl, 439
 - SetAllPrintLevels, 439
 - SetPrintLevel, 438
- Ipopt::Journalist, 440
 - ~Journalist, 442
 - AddFileJournal, 443
 - AddJournal, 443

- DeleteAllJournals, [443](#)
- FlushBuffer, [443](#)
- GetJournal, [443](#)
- Journalist, [442](#)
- journals_, [443](#)
- operator=, [443](#)
- PrintStringOverLines, [442](#)
- Printf, [442](#)
- PrintfIndented, [442](#)
- ProduceOutput, [442](#)
- VPrintf, [442](#)
- VPrintfIndented, [442](#)
- Ipopt::LeastSquareMultipliers, [444](#)
 - ~LeastSquareMultipliers, [445](#)
 - augsysolver_, [445](#)
 - CalculateMultipliers, [445](#)
 - InitializImpl, [445](#)
 - LeastSquareMultipliers, [445](#)
 - operator=, [445](#)
- Ipopt::LimMemQuasiNewtonUpdater, [446](#)
 - ~LimMemQuasiNewtonUpdater, [451](#)
 - AugmentDenseVector, [452](#)
 - AugmentLMatrix, [452](#)
 - AugmentMultiVector, [452](#)
 - AugmentSTDRSMatrix, [452](#)
 - AugmentSdotSMatrix, [452](#)
 - B0_, [456](#)
 - B0_old_, [458](#)
 - CheckSkippingBFGS, [451](#)
 - curr_DR_x_, [455](#)
 - curr_DR_x_tag_, [455](#)
 - curr_eta_, [455](#)
 - curr_lm_memory_, [456](#)
 - curr_lm_memory_old_, [457](#)
 - curr_red_DR_x_, [455](#)
 - D_, [456](#)
 - D_old_, [458](#)
 - DRS_, [457](#)
 - DRS_old_, [459](#)
 - h_space_, [454](#)
 - InitializImpl, [451](#)
 - L_, [456](#)
 - L_old_, [458](#)
 - LMInitialization, [450](#)
 - LMUpdateType, [450](#)
 - last_eta_, [455](#)
 - last_grad_f_, [457](#)
 - last_jac_c_, [457](#)
 - last_jac_d_, [457](#)
 - last_x_, [457](#)
 - LimMemQuasiNewtonUpdater, [451](#)
 - limited_memory_init_val_, [454](#)
 - limited_memory_initialization_, [454](#)
 - limited_memory_max_history_, [454](#)
 - limited_memory_max_skipping_, [454](#)
 - limited_memory_special_for_resto_, [455](#)
 - limited_memory_update_type_, [454](#)
 - lm_skipped_iter_, [455](#)
 - operator=, [451](#)
 - RecalcD, [453](#)
 - RecalcL, [453](#)
 - RecalcY, [453](#)
 - RegisterOptions, [451](#)
 - ReleaseInternalDataBackup, [454](#)
 - RestoreInternalDataBackup, [454](#)
 - S_, [456](#)
 - S_old_, [457](#)
 - STDRS_, [457](#)
 - STDRS_old_, [459](#)
 - SdotS_, [457](#)
 - SdotS_old_, [458](#)
 - SdotS_uptodate_, [457](#)
 - SdotS_uptodate_old_, [458](#)
 - SetW, [454](#)
 - ShiftDenseVector, [453](#)
 - ShiftLMatrix, [453](#)
 - ShiftMultiVector, [452](#)
 - ShiftSTDRSMatrix, [453](#)
 - ShiftSdotSMatrix, [453](#)
 - sigma_, [456](#)
 - sigma_old_, [458](#)
 - sigma_safe_max_, [455](#)
 - sigma_safe_min_, [455](#)
 - SplitEigenvalues, [453](#)
 - StoreInternalDataBackup, [454](#)
 - U_, [456](#)
 - U_old_, [458](#)
 - update_for_resto_, [455](#)
 - UpdateHessian, [451](#)
 - UpdateInternalData, [452](#)
 - V_, [456](#)
 - V_old_, [458](#)
 - Y_, [456](#)
 - Y_old_, [458](#)
 - Ypart_, [456](#)
 - Ypart_old_, [458](#)
- Ipopt::LineSearch, [459](#)
 - ~LineSearch, [460](#)
 - ActivateFallbackMechanism, [461](#)
 - CheckSkippedLineSearch, [461](#)
 - FindAcceptableTrialPoint, [460](#)
 - LineSearch, [460](#)
 - operator=, [461](#)
 - Reset, [461](#)
 - SetRigorousLineSearch, [461](#)
- Ipopt::LoqoMuOracle, [461](#)
 - ~LoqoMuOracle, [462](#)
 - CalculateMu, [463](#)

- InitializeImpl, 463
- LoqoMuOracle, 462, 463
- operator=, 463
- Ipopt::LowRankAugSystemSolver, 463
 - ~LowRankAugSystemSolver, 466
 - aug_system_solver_, 467
 - AugmentedSystemRequiresChange, 467
 - compound_sol_vecspace_, 469
 - d_c_tag_, 468
 - d_d_tag_, 469
 - d_s_tag_, 468
 - d_x_tag_, 468
 - delta_c_, 468
 - delta_d_, 469
 - delta_s_, 468
 - delta_x_, 468
 - first_call_, 469
 - IncreaseQuality, 467
 - InitializeImpl, 466
 - J1_, 469
 - J2_, 469
 - j_c_tag_, 468
 - j_d_tag_, 468
 - LowRankAugSystemSolver, 466
 - num_neg_evals_, 469
 - NumberOfNegEvals, 466
 - operator=, 467
 - ProvidesInertia, 466
 - Solve, 466
 - SolveMultiVector, 467
 - UpdateFactorization, 467
 - Utilde2_, 469
 - Vtilde1_, 469
 - w_factor_, 468
 - w_tag_, 467
 - Wdiag_, 469
- Ipopt::LowRankSSAugSystemSolver, 470
 - ~LowRankSSAugSystemSolver, 472
 - aug_system_solver_, 474
 - AugmentedSystemRequiresChange, 474
 - D_c_ext_, 476
 - d_c_tag_, 475
 - d_d_tag_, 475
 - d_s_tag_, 474
 - d_x_tag_, 474
 - delta_c_, 475
 - delta_d_, 475
 - delta_s_, 475
 - delta_x_, 474
 - expanded_vu_, 476
 - first_call_, 475
 - IncreaseQuality, 473
 - InitializeImpl, 473
 - J_c_ext_, 476
 - j_c_tag_, 475
 - j_d_tag_, 475
 - LowRankSSAugSystemSolver, 472, 473
 - max_rank_, 474
 - negEvalsCorrection_, 476
 - num_neg_evals_, 476
 - NumberOfNegEvals, 473
 - operator=, 473
 - ProvidesInertia, 473
 - Solve, 473
 - UpdateExtendedData, 473
 - w_factor_, 474
 - w_tag_, 474
 - Wdiag_, 475
 - y_c_ext_space_, 476
- Ipopt::LowRankUpdateSymMatrix, 476
 - ~LowRankUpdateSymMatrix, 478
 - ComputeColAMaxImpl, 480
 - ComputeRowAMaxImpl, 480
 - D_, 480
 - GetDiag, 479
 - GetU, 479
 - GetV, 479
 - HasValidNumbersImpl, 480
 - LowRankUpdateSymMatrix, 478, 479
 - LowRankVectorSpace, 479
 - MultVectorImpl, 480
 - operator=, 480
 - owner_space_, 480
 - P_LowRank, 479
 - PrintImpl, 480
 - ReducedDiag, 480
 - SetDiag, 479
 - SetU, 479
 - SetV, 479
 - U_, 481
 - V_, 481
- Ipopt::LowRankUpdateSymMatrixSpace, 481
 - ~LowRankUpdateSymMatrixSpace, 482
 - LowRankUpdateSymMatrixSpace, 482
 - LowRankVectorSpace, 483
 - lowrank_vector_space_, 483
 - MakeNewLowRankUpdateSymMatrix, 483
 - MakeNewSymMatrix, 483
 - operator=, 483
 - P_LowRank, 483
 - P_LowRank_, 483
 - reduced_diag_, 483
 - ReducedDiag, 483
- Ipopt::Ma27TSolverInterface, 484
 - ~Ma27TSolverInterface, 487
 - a_, 491
 - Backsolve, 488
 - cntl_, 490

- dim_, 489
- Factorization, 488
- GetValuesArrayPtr, 487
- icntl_, 490
- ignore_singularity_, 490
- ikeep_, 490
- IncreaseQuality, 488
- InitializeImpl, 487
- InitializeStructure, 487
- initialized_, 489
- iw_, 490
- la_, 491
- la_increase_, 491
- la_init_factor_, 490
- liw_, 490
- liw_increase_, 491
- liw_init_factor_, 489
- Ma27TSolverInterface, 487
- MatrixFormat, 488
- maxfrt_, 491
- meminc_factor_, 490
- MultiSolve, 487
- negevals_, 489
- nonzeros_, 489
- nsteps_, 491
- NumberOfNegEVals, 487
- operator=, 488
- pivotl_, 489
- pivotl_changed_, 489
- pivotlmax_, 489
- ProvidesInertia, 488
- refactorize_, 489
- RegisterOptions, 488
- skip_inertia_check_, 490
- SymbolicFactorization, 488
- warm_start_same_structure_, 490
- Ipopt::Ma28TDependencyDetector, 491
 - ~Ma28TDependencyDetector, 493
 - DetermineDependentRows, 493
 - InitializeImpl, 493
 - jnlst_, 493
 - ma28_pivotl_, 494
 - Ma28TDependencyDetector, 493
 - operator=, 493
 - RegisterOptions, 493
- Ipopt::Ma57TSolverInterface, 494
 - ~Ma57TSolverInterface, 496
 - a_, 500
 - Backsolve, 498
 - dim_, 498
 - Factorization, 498
 - GetValuesArrayPtr, 497
 - IncreaseQuality, 497
 - InitializeImpl, 497
 - InitializeStructure, 497
 - initialized_, 499
 - ma57_pre_alloc_, 499
 - Ma57TSolverInterface, 496
 - MatrixFormat, 498
 - MultiSolve, 497
 - negevals_, 498
 - nonzeros_, 498
 - NumberOfNegEVals, 497
 - operator=, 498
 - pivotl_, 499
 - pivotl_changed_, 499
 - pivotlmax_, 499
 - ProvidesInertia, 497
 - refactorize_, 499
 - RegisterOptions, 498
 - SymbolicFactorization, 498
 - warm_start_same_structure_, 499
 - wd_cntl_, 499
 - wd_fact_, 500
 - wd_icntl_, 499
 - wd_ifact_, 500
 - wd_info_, 499
 - wd_iwork_, 500
 - wd_keep_, 500
 - wd_lfact_, 500
 - wd_lifact_, 500
 - wd_lkeep_, 500
 - wd_rinfo_, 499
- Ipopt::Ma77SolverInterface, 507
 - ~Ma77SolverInterface, 509
 - control_, 512
 - DetermineDependentRows, 511
 - GetValuesArrayPtr, 510
 - IncreaseQuality, 510
 - InitializeImpl, 509
 - InitializeStructure, 510
 - keep_, 511
 - Ma77SolverInterface, 509
 - MatrixFormat, 511
 - MultiSolve, 510
 - ndim_, 511
 - NumberOfNegEVals, 510
 - numneg_, 511
 - order_opts, 509
 - ordering_, 512
 - pivotl_changed_, 511
 - ProvidesDegeneracyDetection, 511
 - ProvidesInertia, 511
 - RegisterOptions, 509
 - umax_, 512
 - val_, 511
- Ipopt::Ma86SolverInterface, 515
 - ~Ma86SolverInterface, 518

- control_, 520
- DetermineDependentRows, 520
- GetValuesArrayPtr, 518
- IncreaseQuality, 519
- InitializeImpl, 518
- InitializeStructure, 518
- keep_, 520
- Ma86SolverInterface, 518
- MatrixFormat, 520
- MultiSolve, 519
- ndim_, 520
- NumberOfNegEvals, 519
- numneg_, 520
- order_, 520
- order_opts, 518
- ordering_, 521
- pivotl_changed_, 520
- ProvidesDegeneracyDetection, 520
- ProvidesInertia, 519
- RegisterOptions, 518
- umax_, 520
- val_, 520
- Ipopt::Ma97SolverInterface, 525
 - ~Ma97SolverInterface, 528
 - akeep_, 531
 - control_, 531
 - current_level_, 532
 - DetermineDependentRows, 530
 - dump_, 532
 - fctidx_, 531
 - fkeep_, 531
 - GetValuesArrayPtr, 529
 - IncreaseQuality, 530
 - InitializeImpl, 529
 - InitializeStructure, 529
 - Ma97SolverInterface, 528
 - MatrixFormat, 530
 - MultiSolve, 529
 - ndim_, 530
 - NumberOfNegEvals, 529
 - numdelay_, 531
 - numneg_, 531
 - order_opts, 528
 - ordering_, 531
 - pivotl_changed_, 531
 - ProvidesDegeneracyDetection, 530
 - ProvidesInertia, 530
 - RegisterOptions, 529
 - rescale_, 531
 - scale_opts, 528
 - ScaleNameToNum, 530
 - scaling_, 531
 - scaling_type_, 531
 - scaling_val_, 532
 - switch_, 531
 - umax_, 531
 - val_, 531
- Ipopt::Matrix, 532
 - ~Matrix, 535
 - AddMSinvZ, 536
 - AddMSinvZImpl, 537
 - cached_valid_, 539
 - ComputeColAMax, 536
 - ComputeColAMaxImpl, 538
 - ComputeRowAMax, 536
 - ComputeRowAMaxImpl, 538
 - HasValidNumbers, 536
 - HasValidNumbersImpl, 538
 - Matrix, 535
 - MultVector, 536
 - MultVectorImpl, 537
 - NCols, 536
 - NRows, 536
 - operator=, 538
 - owner_space_, 539
 - OwnerSpace, 537
 - Print, 537
 - PrintImpl, 538
 - SinvBlrmZMTdBr, 536
 - SinvBlrmZMTdBrImpl, 537
 - TransMultVector, 536
 - TransMultVectorImpl, 537
 - valid_cache_tag_, 539
- Ipopt::MatrixSpace, 539
 - ~MatrixSpace, 541
 - IsMatrixFromSpace, 541
 - MakeNew, 541
 - MatrixSpace, 541
 - NCols, 541
 - nCols_, 542
 - NRows, 541
 - nRows_, 541
 - operator=, 541
- Ipopt::Mc19TSymScalingMethod, 542
 - ~Mc19TSymScalingMethod, 543
 - ComputeSymTScalingFactors, 543
 - InitializeImpl, 543
 - Mc19TSymScalingMethod, 543
 - operator=, 543
- Ipopt::MinC_1NrmRestorationPhase, 546
 - ~MinC_1NrmRestorationPhase, 548
 - bound_mult_reset_threshold_, 549
 - ComputeBoundMultiplierStep, 548
 - constr_mult_reset_threshold_, 549
 - constr_viol_tol_, 549
 - count_restorations_, 549
 - eq_mult_calculator_, 549
 - expect_infeasible_problem_, 549

- InitializeImpl, 548
- MinC_1NrmRestorationPhase, 548
- operator=, 548
- PerformRestoration, 548
- RegisterOptions, 548
- resto_alg_, 549
- resto_failure_feasibility_threshold_, 549
- resto_options_, 549
- Ipopt::MonotoneMuUpdate, 550
 - ~MonotoneMuUpdate, 551
 - barrier_tol_factor_, 552
 - CalcNewMuAndTau, 552
 - compl_inf_tol_, 553
 - first_iter_resto_, 553
 - InitializeImpl, 552
 - initialized_, 553
 - linesearch_, 553
 - MonotoneMuUpdate, 551
 - mu_allow_fast_monotone_decrease_, 552
 - mu_init_, 552
 - mu_linear_decrease_factor_, 552
 - mu_superlinear_decrease_power_, 552
 - mu_target_, 553
 - operator=, 552
 - RegisterOptions, 552
 - tau_min_, 552
 - UpdateBarrierParameter, 552
- Ipopt::MuOracle, 566
 - ~MuOracle, 567
 - CalculateMu, 568
 - InitializeImpl, 568
 - MuOracle, 567
 - operator=, 568
- Ipopt::MuUpdate, 568
 - ~MuUpdate, 569
 - InitializeImpl, 569
 - MuUpdate, 569
 - operator=, 570
 - UpdateBarrierParameter, 570
- Ipopt::MultiVectorMatrix, 553
 - ~MultiVectorMatrix, 555
 - AddOneMultiVectorMatrix, 556
 - AddRightMultMatrix, 556
 - ColVectorSpace, 557
 - ComputeColAMaxImpl, 557
 - ComputeRowAMaxImpl, 557
 - const_vecs_, 558
 - ConstVec, 558
 - FillWithNewVectors, 556
 - GetVector, 556
 - GetVectorNonConst, 556
 - IsValidNumbersImpl, 557
 - LRMultVector, 556
 - MakeNewMultiVectorMatrix, 556
 - MultVectorImpl, 557
 - MultiVectorMatrix, 555
 - MultiVectorMatrixOwnerSpace, 557
 - non_const_vecs_, 558
 - operator=, 558
 - owner_space_, 558
 - PrintImpl, 557
 - ScaleColumns, 556
 - ScaleRows, 556
 - SetVector, 556
 - SetVectorNonConst, 556
 - TransMultVectorImpl, 557
 - Vec, 558
- Ipopt::MultiVectorMatrixSpace, 558
 - ~MultiVectorMatrixSpace, 559
 - ColVectorSpace, 560
 - MakeNew, 560
 - MakeNewMultiVectorMatrix, 559
 - MultiVectorMatrixSpace, 559
 - vec_space_, 560
- Ipopt::MumpsSolverInterface, 560
 - ~MumpsSolverInterface, 563
 - DetermineDependentRows, 564
 - Factorization, 564
 - GetValuesArrayPtr, 563
 - have_symbolic_factorization_, 566
 - IncreaseQuality, 563
 - InitializeImpl, 563
 - InitializeStructure, 563
 - initialized_, 565
 - MatrixFormat, 564
 - mem_percent_, 565
 - MultiSolve, 563
 - mumps_dep_tol_, 566
 - mumps_permuting_scaling_, 565
 - mumps_pivot_order_, 566
 - mumps_ptr_, 565
 - mumps_scaling_, 566
 - MumpsSolverInterface, 563
 - negevals_, 565
 - NumberOfNegEvals, 563
 - operator=, 564
 - pivtol_, 565
 - pivtol_changed_, 565
 - pivtolmax_, 565
 - ProvidesDegeneracyDetection, 564
 - ProvidesInertia, 564
 - refactorize_, 565
 - RegisterOptions, 564
 - Solve, 564
 - SymbolicFactorization, 564
 - warm_start_same_structure_, 566
- Ipopt::NLP, 570
 - ~NLP, 572

- Eval_c, 573
- Eval_d, 573
- Eval_f, 573
- Eval_grad_f, 573
- Eval_h, 574
- Eval_jac_c, 573
- Eval_jac_d, 573
- FinalizeSolution, 574
- GetBoundsInformation, 573
- GetQuasiNewtonApproximationSpaces, 574
- GetScalingParameters, 574
- GetSpaces, 572
- GetStartingPoint, 573
- GetWarmStartIterate, 573
- IntermediateCallBack, 574
- NLP, 572
- operator=, 575
- ProcessOptions, 572
- Ipopt::NLPBoundsRemover, 575
 - ~NLPBoundsRemover, 577
 - allow_twosided_inequalities_, 581
 - d_space_orig_, 580
 - Eval_c, 579
 - Eval_d, 579
 - Eval_f, 578
 - Eval_grad_f, 578
 - Eval_h, 579
 - Eval_jac_c, 579
 - Eval_jac_d, 579
 - FinalizeSolution, 579
 - GetBoundsInformation, 578
 - GetQuasiNewtonApproximationSpaces, 580
 - GetScalingParameters, 579
 - GetSpaces, 578
 - GetStartingPoint, 578
 - GetWarmStartIterate, 578
 - IntermediateCallBack, 579
 - NLPBoundsRemover, 577
 - nlp, 580
 - nlp_, 580
 - operator=, 580
 - ProcessOptions, 578
 - Px_l_orig_, 580
 - Px_u_orig_, 580
- Ipopt::NLPScalingObject, 581
 - ~NLPScalingObject, 584
 - apply_grad_obj_scaling, 587
 - apply_grad_obj_scaling_NonConst, 586
 - apply_hessian_scaling, 586
 - apply_jac_c_scaling, 586
 - apply_jac_d_scaling, 586
 - apply_obj_scaling, 584
 - apply_vector_scaling_c, 585
 - apply_vector_scaling_c_NonConst, 585
 - apply_vector_scaling_d, 585
 - apply_vector_scaling_d_LU, 586
 - apply_vector_scaling_d_NonConst, 585
 - apply_vector_scaling_x, 584
 - apply_vector_scaling_x_NonConst, 584
 - apply_vector_scaling_d, 585
 - apply_vector_scaling_d_LU, 586
 - apply_vector_scaling_d_NonConst, 586
 - apply_vector_scaling_x, 584
 - apply_vector_scaling_x_NonConst, 584
 - DetermineScaling, 587
 - have_c_scaling, 587
 - have_d_scaling, 587
 - have_x_scaling, 587
 - Initialize, 584
 - InitializeImpl, 587
 - Jnlst, 587
 - jnlst_, 588
 - NLPScalingObject, 584
 - operator=, 588
 - unapply_grad_obj_scaling, 587
 - unapply_grad_obj_scaling_NonConst, 587
 - unapply_obj_scaling, 584
 - unapply_vector_scaling_c, 585
 - unapply_vector_scaling_c_NonConst, 585
 - unapply_vector_scaling_d, 585
 - unapply_vector_scaling_d_LU, 586
 - unapply_vector_scaling_d_NonConst, 585
 - unapply_vector_scaling_x, 584
 - unapply_vector_scaling_x_NonConst, 584
- Ipopt::NoNLPScalingObject, 588
 - ~NoNLPScalingObject, 589
 - DetermineScalingParametersImpl, 589
 - NoNLPScalingObject, 589
 - operator=, 589
- Ipopt::Observer, 590
 - ~Observer, 591
 - NotifyType, 591
 - Observer, 591
 - operator=, 592
 - ProcessNotification, 592
 - RecieveNotification, 592
 - RequestAttach, 592
 - RequestDetach, 592
 - Subject, 592
 - subjects_, 592
- Ipopt::OptimalityErrorConvergenceCheck, 593
 - ~OptimalityErrorConvergenceCheck, 595
 - acceptable_compl_inf_tol_, 596
 - acceptable_constr_viol_tol_, 596
 - acceptable_counter_, 597
 - acceptable_dual_inf_tol_, 596
 - acceptable_iter_, 596
 - acceptable_obj_change_tol_, 597
 - acceptable_tol_, 596
 - CheckConvergence, 595

- compl_inf_tol_, 596
- constr_viol_tol_, 596
- curr_obj_val_, 597
- CurrentIsAcceptable, 595
- diverging_iterates_tol_, 597
- dual_inf_tol_, 596
- InitializeImpl, 595
- last_obj_val_, 597
- last_obj_val_iter_, 597
- max_cpu_time_, 597
- max_iterations_, 596
- mu_target_, 597
- operator=, 595
- OptimalityErrorConvergenceCheck, 595
- RegisterOptions, 595
- Ipopt::OptionsList, 597
 - ~OptionsList, 600
 - clear, 600
 - find_tag, 601
 - GetBoolValue, 601
 - GetEnumValue, 601
 - GetIntegerValue, 601
 - GetNumericValue, 601
 - GetStringValue, 601
 - jnlst_, 602
 - lowercase, 601
 - lowercase_buffer_, 602
 - operator=, 600
 - options_, 602
 - OptionsList, 600
 - PrintList, 601
 - PrintUserOptions, 601
 - ReadFromStream, 601
 - readnexttoken, 601
 - reg_options_, 602
 - SetIntegerValue, 600
 - SetIntegerValueIfUnset, 601
 - SetJournalist, 600
 - SetNumericValue, 600
 - SetNumericValueIfUnset, 600
 - SetRegisteredOptions, 600
 - SetStringValue, 600
 - SetStringValueIfUnset, 600
 - will_allow_clobber, 601
- Ipopt::OptionsList::OptionValue, 602
 - ~OptionValue, 603
 - allow_clobber_, 604
 - AllowClobber, 604
 - Counter, 604
 - counter_, 604
 - dont_print_, 605
 - DontPrint, 604
 - GetValue, 604
 - initialized_, 604
 - operator=, 604
 - OptionValue, 603
 - Value, 604
 - value_, 604
- Ipopt::OrigIpoptNLP, 605
 - ~OrigIpoptNLP, 610
 - AdjustVariableBounds, 613
 - bound_relax_factor_, 619
 - c, 611
 - c_cache_, 617
 - c_eval_time, 615
 - c_eval_time_, 620
 - c_evals, 614
 - c_evals_, 620
 - c_space_, 616
 - check_derivatives_for_naninf_, 619
 - d, 611
 - d_L, 612
 - d_L_, 618
 - d_U, 613
 - d_U_, 618
 - d_cache_, 617
 - d_eval_time, 615
 - d_eval_time_, 620
 - d_evals, 614
 - d_evals_, 620
 - d_l_space_, 616
 - d_space_, 616
 - d_u_space_, 616
 - f, 611
 - f_cache_, 617
 - f_eval_time, 615
 - f_eval_time_, 620
 - f_evals, 613
 - f_evals_, 620
 - FinalizeSolution, 614
 - get_unscaled_x, 615
 - GetSpaces, 613
 - GetWarmStartIterate, 611
 - grad_f, 611
 - grad_f_cache_, 617
 - grad_f_eval_time, 615
 - grad_f_eval_time_, 620
 - grad_f_evals, 614
 - grad_f_evals_, 620
 - h, 612
 - h_cache_, 617
 - h_eval_time, 615
 - h_eval_time_, 621
 - h_evals, 614
 - h_evals_, 620
 - h_space_, 617
 - hessian_approximation_, 619
 - hessian_approximation_space_, 619

hessian_constant_, 619
HessianMatrixSpace, 613
honor_original_bounds_, 619
Initialize, 610
InitializeStructures, 610
initialized_, 620
IntermediateCallBack, 614
jac_c, 611
jac_c_cache_, 617
jac_c_constant_, 619
jac_c_eval_time, 615
jac_c_eval_time_, 620
jac_c_evals, 614
jac_c_evals_, 620
jac_c_space_, 616
jac_d, 611
jac_d_cache_, 617
jac_d_constant_, 619
jac_d_eval_time, 615
jac_d_eval_time_, 620
jac_d_evals, 614
jac_d_evals_, 620
jac_d_space_, 617
jnlst_, 616
nlp, 614
nlp_, 616
operator=, 615
orig_x_L_, 618
orig_x_U_, 619
OrigIpoptNLP, 610
Pd_L, 613
Pd_L_, 618
Pd_U, 613
Pd_U_, 618
pd_l_space_, 616
pd_u_space_, 616
PrintTimingStatistics, 615
Px_L, 612
Px_L_, 618
Px_U, 612
Px_U_, 618
px_l_space_, 616
px_u_space_, 616
RegisterOptions, 614
relax_bounds, 615
ResetTimes, 615
scaled_h_space_, 617
scaled_jac_c_space_, 617
scaled_jac_d_space_, 617
TotalFunctionEvaluationCpuTime, 615
TotalFunctionEvaluationSysTime, 615
TotalFunctionEvaluationWallclockTime, 615
uninitialized_h, 612
unscaled_x_cache_, 618

warm_start_same_structure_, 619
x_L, 612
x_L_, 618
x_U, 612
x_U_, 618
x_l_space_, 616
x_space_, 616
x_u_space_, 616
Ipopt::OrigIterationOutput, 621
~OrigIterationOutput, 622
inf_pr_output_, 623
InitializeImpl, 622
operator=, 623
OrigIterationOutput, 622
print_frequency_iter_, 623
print_frequency_time_, 623
print_info_string_, 623
RegisterOptions, 623
WriteOutput, 622
Ipopt::PDFullSpaceSolver, 631
~PDFullSpaceSolver, 633
augSysSolver_, 634
augsys_improved_, 634
ComputeResidualRatio, 634
ComputeResiduals, 634
dummy_cache_, 634
InitializeImpl, 633
max_refinement_steps_, 635
min_refinement_steps_, 634
neg_curv_test_tol_, 635
operator=, 633
PDFullSpaceSolver, 633
perturbHandler_, 634
RegisterOptions, 633
residual_improvement_factor_, 635
residual_ratio_max_, 635
residual_ratio_singular_, 635
SinvBlrmZPTdBr, 634
Solve, 633
SolveOnce, 633
Ipopt::PDPerturbationHandler, 635
~PDPerturbationHandler, 639
ConsiderNewSystem, 639
CurrentPerturbation, 640
degen_iters_, 641
degen_iters_max_, 642
DegenType, 638
delta_c_curr_, 641
delta_c_last_, 640
delta_cd, 640
delta_cd_exp_, 642
delta_cd_val_, 642
delta_d_curr_, 641
delta_d_last_, 640

- delta_s_curr_, 641
- delta_s_last_, 640
- delta_x_curr_, 640
- delta_x_last_, 640
- delta_xs_dec_fact_, 642
- delta_xs_first_inc_fact_, 642
- delta_xs_inc_fact_, 642
- delta_xs_init_, 642
- delta_xs_max_, 641
- delta_xs_min_, 641
- finalize_test, 640
- get_deltas_for_wrong_inertia, 640
- get_deltas_for_wrong_inertia_called_, 641
- hess_degenerate_, 641
- InitializeImpl, 639
- jac_degenerate_, 641
- operator=, 640
- PDPerturbationHandler, 639
- perturb_always_cd_, 642
- PerturbForSingularity, 639
- PerturbForWrongInertia, 639
- RegisterOptions, 640
- reset_last_, 642
- test_status_, 641
- TrialStatus, 638
- Ipopt::PDSearchDirCalculator, 643
 - ~PDSearchDirCalculator, 644
 - ComputeSearchDirection, 645
 - fast_step_computation_, 645
 - InitializeImpl, 644
 - mehrotra_algorithm_, 645
 - operator=, 645
 - PDSearchDirCalculator, 644
 - PDSolver, 645
 - pd_solver_, 645
 - RegisterOptions, 645
- Ipopt::PDSystemSolver, 645
 - ~PDSystemSolver, 647
 - InitializeImpl, 647
 - operator=, 647
 - PDSystemSolver, 647
 - Solve, 647
- Ipopt::PardisoSolverInterface, 623
 - ~PardisoSolverInterface, 627
 - a_, 628
 - DPARM_, 630
 - debug_cnt_, 630
 - debug_last_iter_, 630
 - dim_, 628
 - Factorization, 628
 - GetValuesArrayPtr, 627
 - have_symbolic_factorization_, 629
 - IPARM_, 630
 - IncreaseQuality, 627
 - InitializeImpl, 627
 - InitializeStructure, 627
 - initialized_, 629
 - MAXFCT_, 630
 - MNUM_, 630
 - MSGGLVL_, 630
 - MTYPE_, 630
 - match_strat_, 629
 - MatrixFormat, 628
 - MultiSolve, 627
 - negevals_, 629
 - nonzeros_, 628
 - NumberOfNegEvals, 627
 - operator=, 628
 - PT_, 629
 - pardiso_iterative_, 629
 - pardiso_max_droptol_corrections_, 629
 - pardiso_repeated_perturbation_means_singular_, 629
 - PardisoMatchingStrategy, 626
 - PardisoSolverInterface, 627
 - ProvidesInertia, 628
 - RegisterOptions, 628
 - skip_inertia_check_, 629
 - Solve, 628
 - SymbolicFactorization, 628
- Ipopt::PenaltyLSAcceptor, 648
 - ~PenaltyLSAcceptor, 651
 - CalcPred, 653
 - CalculateAlphaMin, 651
 - CheckAcceptabilityOfTrialPoint, 651
 - eta_, 653
 - InitThisLineSearch, 651
 - InitializeImpl, 651
 - IsAcceptableToCurrentIterate, 652
 - kappa_soc_, 653
 - last_nu_, 654
 - max_soc_, 653
 - nu_, 654
 - nu_inc_, 653
 - nu_init_, 653
 - operator=, 652
 - pd_solver_, 655
 - PenaltyLSAcceptor, 651
 - PrepareRestoPhaseStart, 651
 - reference_JacC_delta_, 654
 - reference_JacD_delta_, 654
 - reference_barr_, 653
 - reference_dWd_, 654
 - reference_gradBarrTDelta_, 653
 - reference_pred_, 654
 - reference_theta_, 653
 - RegisterOptions, 652
 - Reset, 651

resto_pred_, 654
 rho_, 653
 StartWatchDog, 652
 StopWatchDog, 652
 TryCorrector, 652
 TrySecondOrderCorrection, 652
 UpdateForNextIteration, 652
 watchdog_barr_, 654
 watchdog_pred_, 654
 watchdog_theta_, 654
 Ipopt::PiecewisePenEntry, 658
 barrier_obj, 658
 infeasible, 658
 pen_r, 658
 Ipopt::PiecewisePenalty, 655
 ~PiecewisePenalty, 656
 Acceptable, 657
 AddEntry, 657
 BiggestBarr, 657
 Clear, 657
 dim_, 657
 InitPiecewisePenaltyList, 657
 IsPiecewisePenaltyListEmpty, 657
 max_piece_number_, 658
 min_piece_penalty_, 657
 operator=, 657
 PiecewisePenalty, 656
 PiecewisePenalty_list_, 658
 Print, 657
 ResetList, 657
 UpdateEntry, 657
 Ipopt::PointPerturber, 659
 ~PointPerturber, 660
 MakeNewPerturbedPoint, 660
 operator=, 660
 pert_dir_, 660
 PointPerturber, 660
 ref_point_, 660
 Ipopt::ProbingMuOracle, 662
 ~ProbingMuOracle, 663
 CalculateAffineMu, 664
 CalculateMu, 664
 InitializeImpl, 664
 operator=, 664
 pd_solver_, 664
 ProbingMuOracle, 663
 RegisterOptions, 664
 sigma_max_, 664
 Ipopt::QualityFunctionMuOracle, 664
 ~QualityFunctionMuOracle, 668
 BalancingTermEnum, 668
 CalculateMu, 669
 CalculateQualityFunction, 669
 CentralityEnum, 668
 count_qf_evals_, 672
 curr_c_amax_, 673
 curr_c_asum_, 673
 curr_c_nrm2_, 673
 curr_d_minus_s_amax_, 673
 curr_d_minus_s_asum_, 673
 curr_d_minus_s_nrm2_, 673
 curr_grad_lag_s_amax_, 673
 curr_grad_lag_s_asum_, 673
 curr_grad_lag_s_nrm2_, 673
 curr_grad_lag_x_amax_, 673
 curr_grad_lag_x_asum_, 672
 curr_grad_lag_x_nrm2_, 673
 curr_slack_s_L_, 672
 curr_slack_s_U_, 672
 curr_slack_x_L_, 672
 curr_slack_x_U_, 672
 curr_v_L_, 672
 curr_v_U_, 672
 curr_z_L_, 672
 curr_z_U_, 672
 InitializeImpl, 669
 initialized_, 672
 n_comp_, 672
 n_dual_, 672
 n_pri_, 672
 NormEnum, 668
 operator=, 669
 pd_solver_, 670
 PerformGoldenSection, 669
 quality_function_balancing_term_, 670
 quality_function_centrality_, 670
 quality_function_max_section_steps_, 670
 quality_function_norm_, 670
 quality_function_section_qf_tol_, 670
 quality_function_section_sigma_tol_, 670
 QualityFunctionMuOracle, 668
 RegisterOptions, 669
 ScaleSigma, 669
 sigma_max_, 670
 sigma_min_, 670
 tmp_slack_s_L_, 671
 tmp_slack_s_U_, 671
 tmp_slack_x_L_, 671
 tmp_slack_x_U_, 671
 tmp_step_s_L_, 671
 tmp_step_s_U_, 671
 tmp_step_v_L_, 671
 tmp_step_v_U_, 671
 tmp_step_x_L_, 670
 tmp_step_x_U_, 670
 tmp_step_z_L_, 671
 tmp_step_z_U_, 671
 tmp_v_L_, 671

- tmp_v_U_, 671
- tmp_z_L_, 671
- tmp_z_U_, 671
- UnscaleSigma, 669
- Ipopt::ReferencedObject, 673
 - ~ReferencedObject, 676
 - AddRef, 676
 - reference_count_, 677
 - ReferenceCount, 676
 - ReferencedObject, 676
 - ReleaseRef, 676
- Ipopt::Referencer, 677
- Ipopt::RegisteredOption, 678
 - ~RegisteredOption, 681
 - AddValidStringSetting, 683
 - Counter, 682
 - counter_, 686
 - default_number_, 686
 - default_string_, 686
 - DefaultInteger, 684
 - DefaultNumber, 684
 - DefaultString, 684
 - DefaultStringAsEnum, 684
 - GetValidStrings, 684
 - has_lower_, 686
 - has_upper_, 686
 - HasLower, 682
 - HasUpper, 683
 - IsValidIntegerSetting, 685
 - IsValidNumberSetting, 684
 - IsValidStringSetting, 685
 - long_description_, 686
 - LongDescription, 681
 - lower_, 686
 - lower_strict_, 686
 - LowerInteger, 683
 - LowerNumber, 682
 - LowerStrict, 682
 - MakeValidLatexNumber, 685
 - MakeValidLatexString, 685
 - MapStringSetting, 685
 - MapStringSettingToEnum, 685
 - Name, 681
 - name_, 685
 - OutputDescription, 685
 - OutputLatexDescription, 685
 - OutputShortDescription, 685
 - RegisteredOption, 681
 - registering_category_, 686
 - RegisteringCategory, 682
 - SetDefaultInteger, 684
 - SetDefaultNumber, 684
 - SetDefaultString, 684
 - SetLongDescription, 682
 - SetLowerInteger, 683
 - SetLowerNumber, 683
 - SetName, 681
 - SetRegisteringCategory, 682
 - SetShortDescription, 681
 - SetType, 682
 - SetUpperInteger, 683
 - SetUpperNumber, 683
 - short_description_, 685
 - ShortDescription, 681
 - string_equal_insensitive, 685
 - Type, 682
 - type_, 686
 - upper_, 686
 - upper_strict_, 686
 - UpperInteger, 683
 - UpperNumber, 683
 - UpperStrict, 683
 - valid_strings_, 686
- Ipopt::RegisteredOption::string_entry, 777
 - description_, 778
 - string_entry, 778
 - value_, 778
- Ipopt::RegisteredOptions, 687
 - ~RegisteredOptions, 689
 - AddBoundedIntegerOption, 691
 - AddBoundedNumberOption, 690
 - AddIntegerOption, 690
 - AddLowerBoundedIntegerOption, 690
 - AddLowerBoundedNumberOption, 690
 - AddNumberOption, 690
 - AddStringOption, 691
 - AddStringOption1, 691
 - AddStringOption10, 692
 - AddStringOption2, 691
 - AddStringOption3, 691
 - AddStringOption4, 691
 - AddStringOption5, 691
 - AddStringOption6, 691
 - AddStringOption7, 691
 - AddStringOption8, 692
 - AddStringOption9, 692
 - AddUpperBoundedIntegerOption, 690
 - AddUpperBoundedNumberOption, 690
 - current_registering_category_, 693
 - GetOption, 692
 - next_counter_, 692
 - OutputLatexOptionDocumentation, 692
 - OutputOptionDocumentation, 692
 - RegOptionsList, 689
 - registered_options_, 693
 - RegisteredOptions, 689
 - RegisteredOptionsList, 692
 - RegisteringCategory, 690

- SetRegisteringCategory, 690
- Ipopt::RestoConvergenceCheck, 693
 - ~RestoConvergenceCheck, 695
 - CheckConvergence, 695
 - first_resto_iter_, 696
 - InitializeImpl, 695
 - kappa_resto_, 696
 - maximum_iters_, 696
 - maximum_resto_iters_, 696
 - operator=, 695
 - orig_constr_viol_tol_, 696
 - RegisterOptions, 695
 - RestoConvergenceCheck, 695
 - SetOrigLSAcceptor, 695
 - successive_resto_iter_, 696
 - TestOrigProgress, 695
- Ipopt::RestoFilterConvergenceCheck, 696
 - ~RestoFilterConvergenceCheck, 698
 - InitializeImpl, 698
 - operator=, 698
 - orig_filter_ls_acceptor_, 699
 - RegisterOptions, 698
 - RestoFilterConvergenceCheck, 698
 - SetOrigLSAcceptor, 698
 - TestOrigProgress, 699
- Ipopt::RestolpoptNLP, 699
 - ~RestolpoptNLP, 704
 - AdjustVariableBounds, 707
 - c, 705
 - c_evals, 708
 - c_evals_, 712
 - c_space_, 709
 - d, 705
 - d_L, 706
 - d_L_, 710
 - d_U, 706
 - d_U_, 711
 - d_evals, 708
 - d_evals_, 712
 - d_l_space_, 710
 - d_space_, 709
 - d_u_space_, 710
 - DR_x, 708
 - DR_x_, 711
 - dr_x_, 711
 - Eta, 708
 - eta_factor_, 711
 - eta_mu_exponent_, 711
 - evaluate_orig_obj_at_resto_trial_, 711
 - f, 705
 - f_evals, 708
 - f_evals_, 712
 - FinalizeSolution, 704
 - GetSpaces, 707
 - GetWarmStartIterate, 704
 - grad_f, 705
 - grad_f_evals, 708
 - grad_f_evals_, 712
 - h, 705
 - h_evals, 708
 - h_evals_, 712
 - h_space_, 710
 - hessian_approximation_, 712
 - HessianMatrixSpace, 707
 - Initialize, 704
 - InitializeStructures, 704
 - initialized_, 712
 - IntermediateCallBack, 707
 - jac_c, 705
 - jac_c_evals, 708
 - jac_c_evals_, 712
 - jac_c_space_, 710
 - jac_d, 705
 - jac_d_evals, 708
 - jac_d_evals_, 712
 - jac_d_space_, 710
 - objective_depends_on_mu, 704
 - operator=, 709
 - orig_ip_cq_, 709
 - orig_ip_data_, 709
 - orig_ip_nlp_, 709
 - OrigIpCq, 708
 - OrigIpData, 707
 - OrigIpNLP, 707
 - Pd_L, 706
 - Pd_L_, 711
 - Pd_U, 707
 - Pd_U_, 711
 - pd_l_space_, 710
 - pd_u_space_, 710
 - Px_L, 706
 - Px_L_, 710
 - Px_U, 706
 - Px_U_, 710
 - px_l_space_, 709
 - px_u_space_, 710
 - RegisterOptions, 709
 - RestolpoptNLP, 704
 - Rho, 708
 - rho_, 711
 - uninitialized_h, 706
 - x_L, 706
 - x_L_, 710
 - x_U, 706
 - x_U_, 710
 - x_l_space_, 709
 - x_ref_, 711
 - x_space_, 709

- x_u_space_, 709
- Ipopt::RestolterateInitializer, 712
 - ~RestolterateInitializer, 714
 - constr_mult_init_max_, 715
 - InitializeImpl, 714
 - operator=, 715
 - RegisterOptions, 714
 - resto_eq_mult_calculator_, 715
 - RestolterateInitializer, 714
 - SetInitialIterates, 714
 - solve_quadratic, 715
- Ipopt::RestolterationOutput, 715
 - ~RestolterationOutput, 717
 - inf_pr_output_, 717
 - InitializeImpl, 717
 - operator=, 717
 - print_frequency_iter_, 718
 - print_frequency_time_, 718
 - print_info_string_, 717
 - resto_orig_iteration_output_, 717
 - RestolterationOutput, 717
 - WriteOutput, 717
- Ipopt::RestoPenaltyConvergenceCheck, 718
 - ~RestoPenaltyConvergenceCheck, 719
 - InitializeImpl, 720
 - operator=, 720
 - orig_penalty_ls_acceptor_, 720
 - RegisterOptions, 720
 - RestoPenaltyConvergenceCheck, 719
 - SetOrigLSAcceptor, 720
 - TestOrigProgress, 720
- Ipopt::RestoRestorationPhase, 722
 - ~RestoRestorationPhase, 724
 - InitializeImpl, 724
 - operator=, 724
 - PerformRestoration, 724
 - RestoRestorationPhase, 724
 - solve_quadratic, 724
- Ipopt::RestorationPhase, 721
 - ~RestorationPhase, 722
 - InitializeImpl, 722
 - operator=, 722
 - PerformRestoration, 722
 - RestorationPhase, 722
- Ipopt::ScaledMatrix, 724
 - ~ScaledMatrix, 726
 - AddMSinvZImpl, 728
 - ColumnScaling, 727
 - ComputeColAMaxImpl, 728
 - ComputeRowAMaxImpl, 728
 - GetUnscaledMatrix, 727
 - GetUnscaledMatrixNonConst, 727
 - IsValidNumbersImpl, 727
 - matrix_, 728
 - MultVectorImpl, 727
 - nonconst_matrix_, 729
 - operator=, 728
 - owner_space_, 729
 - PrintImpl, 728
 - RowScaling, 727
 - ScaledMatrix, 726
 - SetUnscaledMatrix, 727
 - SetUnscaledMatrixNonConst, 727
 - SinvBlrmZMTdBrlImpl, 728
 - TransMultVectorImpl, 727
- Ipopt::ScaledMatrixSpace, 729
 - ~ScaledMatrixSpace, 730
 - column_scaling_, 731
 - ColumnScaling, 731
 - MakeNew, 731
 - MakeNewScaledMatrix, 731
 - operator=, 731
 - row_scaling_, 731
 - RowScaling, 731
 - ScaledMatrixSpace, 730
 - unscaled_matrix_space_, 731
 - UnscaledMatrixSpace, 731
- Ipopt::SearchDirectionCalculator, 732
 - ~SearchDirectionCalculator, 733
 - ComputeSearchDirection, 733
 - InitializeImpl, 733
 - operator=, 733
 - SearchDirectionCalculator, 733
- Ipopt::SlackBasedTSymScalingMethod, 733
 - ~SlackBasedTSymScalingMethod, 734
 - ComputeSymTScalingFactors, 735
 - InitializeImpl, 735
 - operator=, 735
 - SlackBasedTSymScalingMethod, 734, 735
- Ipopt::SmartPtr
 - ~SmartPtr, 739
 - ConstPtr, 741
 - GetRawPtr, 741
 - IsNull, 741
 - IsValid, 741
 - operator<, 741
 - operator*, 739
 - operator->, 739
 - operator=, 740
 - operator==, 740
 - ptr_, 741
 - ReleasePointer_, 740
 - SetFromRawPtr_, 740
 - SetFromSmartPtr_, 740
 - SmartPtr, 739
- Ipopt::SmartPtr< T >, 735
- Ipopt::SolveStatistics, 742
 - ~SolveStatistics, 744

- compl_, 746
- constr_viol_, 746
- dual_inf_, 746
- FinalObjective, 745
- FinalScaledObjective, 745
- Infeasibilities, 745
- IterationCount, 744
- kkt_error_, 747
- num_constr_evals_, 745
- num_constr_jac_evals_, 746
- num_hess_evals_, 746
- num_iters_, 745
- num_obj_evals_, 745
- num_obj_grad_evals_, 745
- NumberOfEvaluations, 744
- obj_val_, 746
- operator=, 745
- scaled_compl_, 746
- scaled_constr_viol_, 746
- scaled_dual_inf_, 746
- scaled_kkt_error_, 747
- scaled_obj_val_, 746
- ScaledInfeasibilities, 745
- SolveStatistics, 744
- total_cpu_time_, 745
- total_sys_time_, 745
- total_wallclock_time_, 745
- TotalCPUTime, 744
- TotalCpuTime, 744
- TotalSysTime, 744
- TotalWallclockTime, 744
- Ipopt::SparseSymLinearSolverInterface, 747
 - ~SparseSymLinearSolverInterface, 750
 - DetermineDependentRows, 752
 - EMatrixFormat, 750
 - GetValuesArrayPtr, 751
 - IncreaseQuality, 752
 - InitializeImpl, 751
 - InitializeStructure, 751
 - MatrixFormat, 752
 - MultiSolve, 751
 - NumberOfNegEVals, 752
 - ProvidesDegeneracyDetection, 752
 - ProvidesInertia, 752
 - SparseSymLinearSolverInterface, 750
- Ipopt::StandardScalingBase, 753
 - ~StandardScalingBase, 755
 - apply_hessian_scaling, 757
 - apply_jac_c_scaling, 757
 - apply_jac_d_scaling, 757
 - apply_obj_scaling, 756
 - apply_vector_scaling_c, 756
 - apply_vector_scaling_c_NonConst, 756
 - apply_vector_scaling_d, 757
- apply_vector_scaling_d_NonConst, 757
- apply_vector_scaling_x, 756
- apply_vector_scaling_x_NonConst, 756
- DetermineScaling, 758
- DetermineScalingParametersImpl, 758
- df_, 759
- dx_, 759
- have_c_scaling, 758
- have_d_scaling, 758
- have_x_scaling, 758
- InitializeImpl, 758
- obj_scaling_factor_, 759
- operator=, 759
- RegisterOptions, 758
- scaled_h_space_, 759
- scaled_jac_c_space_, 759
- scaled_jac_d_space_, 759
- StandardScalingBase, 755
- unapply_obj_scaling, 756
- unapply_vector_scaling_c, 756
- unapply_vector_scaling_c_NonConst, 757
- unapply_vector_scaling_d, 757
- unapply_vector_scaling_d_NonConst, 757
- unapply_vector_scaling_x, 756
- unapply_vector_scaling_x_NonConst, 756
- Ipopt::StdAugSystemSolver, 759
 - ~StdAugSystemSolver, 762
 - augmented_system_, 766
 - augmented_system_space_, 764
 - augmented_vector_space_, 764
 - AugmentedSystemRequiresChange, 763
 - augsys_tag_, 766
 - CreateAugmentedSpace, 763
 - CreateAugmentedSystem, 763
 - d_c_tag_, 765
 - d_d_tag_, 765
 - d_s_tag_, 765
 - d_x_tag_, 765
 - delta_c_, 765
 - delta_d_, 766
 - delta_s_, 765
 - delta_x_, 765
 - diag_space_c_, 764
 - diag_space_d_, 764
 - diag_space_s_, 764
 - diag_space_x_, 764
 - ident_space_ds_, 764
 - IncreaseQuality, 763
 - InitializeImpl, 762
 - j_c_tag_, 765
 - j_d_tag_, 765
 - linsolver_, 764
 - MultiSolve, 762
 - NumberOfNegEVals, 763

- old_w_, 766
- operator=, 763
- ProvidesInertia, 763
- StdAugSystemSolver, 762
- sumsym_space_x_, 764
- w_factor_, 764
- w_tag_, 764
- warm_start_same_structure_, 766
- Ipopt::StdInterfaceTNLP, 766
 - ~StdInterfaceTNLP, 770
 - apply_new_x_, 772
 - eval_f_, 771
 - eval_f_, 773
 - eval_g_, 771
 - eval_g_, 773
 - eval_grad_f_, 771
 - eval_grad_f_, 773
 - eval_h_, 771
 - eval_h_, 774
 - eval_jac_g_, 771
 - eval_jac_g_, 774
 - finalize_solution, 772
 - g_L_, 772
 - g_U_, 772
 - g_scaling_, 774
 - g_sol_, 775
 - get_bounds_info, 770
 - get_nlp_info, 770
 - get_scaling_parameters, 770
 - get_starting_point, 770
 - index_style_, 773
 - intermediate_callback, 771
 - intermediate_cb_, 774
 - jnlst_, 772
 - lambda_sol_, 775
 - n_con_, 772
 - n_var_, 772
 - nele_hess_, 773
 - nele_jac_, 773
 - non_const_x_, 774
 - obj_scaling_, 774
 - obj_sol_, 775
 - operator=, 772
 - start_lam_, 773
 - start_x_, 773
 - start_z_L_, 773
 - start_z_U_, 773
 - StdInterfaceTNLP, 770
 - user_data_, 774
 - x_L_, 772
 - x_U_, 772
 - x_scaling_, 774
 - x_sol_, 774
 - z_L_sol_, 774
 - z_U_sol_, 775
- Ipopt::StreamJournal, 775
 - ~StreamJournal, 776
 - buffer_, 777
 - FlushBufferImpl, 777
 - operator=, 777
 - os_, 777
 - PrintImpl, 777
 - PrintfImpl, 777
 - SetOutputStream, 777
 - StreamJournal, 776
- Ipopt::Subject, 778
 - ~Subject, 780
 - AttachObserver, 780
 - DetachObserver, 780
 - Notify, 780
 - observers_, 781
 - operator=, 781
 - Subject, 780
- Ipopt::SumMatrix, 781
 - ~SumMatrix, 783
 - ComputeColAMaxImpl, 784
 - ComputeRowAMaxImpl, 783
 - factors_, 784
 - GetTerm, 783
 - IsValidNumbersImpl, 783
 - matrices_, 784
 - MultVectorImpl, 783
 - NTerms, 783
 - operator=, 784
 - owner_space_, 784
 - PrintImpl, 784
 - SetTerm, 783
 - SumMatrix, 783
 - TransMultVectorImpl, 783
- Ipopt::SumMatrixSpace, 784
 - ~SumMatrixSpace, 786
 - GetTermSpace, 786
 - MakeNew, 786
 - MakeNewSumMatrix, 786
 - NTerms, 786
 - nterms_, 787
 - operator=, 786
 - SetTermSpace, 786
 - SumMatrixSpace, 786
 - term_spaces_, 787
- Ipopt::SumSymMatrix, 787
 - ~SumSymMatrix, 789
 - ComputeColAMaxImpl, 790
 - ComputeRowAMaxImpl, 789
 - factors_, 790
 - GetTerm, 789
 - IsValidNumbersImpl, 789
 - matrices_, 790

- MultVectorImpl, 789
- NTerms, 789
- operator=, 790
- owner_space_, 790
- PrintImpl, 790
- SetTerm, 789
- SumSymMatrix, 789
- Ipopt::SumSymMatrixSpace, 790
 - ~SumSymMatrixSpace, 791
 - GetTermSpace, 792
 - MakeNewSumSymMatrix, 792
 - MakeNewSymMatrix, 792
 - NTerms, 792
 - nterms_, 792
 - SetTermSpace, 792
 - SumSymMatrixSpace, 791
 - term_spaces_, 792
- Ipopt::SymLinearSolver, 792
 - ~SymLinearSolver, 794
 - IncreaseQuality, 794
 - InitializImpl, 794
 - MultiSolve, 794
 - NumberOfNegEVals, 794
 - ProvidesInertia, 795
 - Solve, 794
 - SymLinearSolver, 794
- Ipopt::SymMatrix, 795
 - ~SymMatrix, 796
 - ComputeColAMaxImpl, 797
 - Dim, 797
 - owner_space_, 797
 - OwnerSymMatrixSpace, 797
 - SymMatrix, 796
 - TransMultVectorImpl, 797
- Ipopt::SymMatrixSpace, 797
 - ~SymMatrixSpace, 799
 - Dim, 799
 - MakeNew, 799
 - MakeNewSymMatrix, 799
 - operator=, 799
 - SymMatrixSpace, 799
- Ipopt::SymScaledMatrix, 800
 - ~SymScaledMatrix, 802
 - ComputeRowAMaxImpl, 803
 - GetUnscaledMatrix, 802
 - GetUnscaledMatrixNonConst, 802
 - IsValidNumbersImpl, 803
 - matrix_, 803
 - MultVectorImpl, 802
 - nonconst_matrix_, 803
 - operator=, 803
 - owner_space_, 803
 - PrintImpl, 803
 - RowColScaling, 802
 - SetUnscaledMatrix, 802
 - SetUnscaledMatrixNonConst, 802
 - SymScaledMatrix, 802
- Ipopt::SymScaledMatrixSpace, 803
 - ~SymScaledMatrixSpace, 805
 - MakeNew, 805
 - MakeNewSymMatrix, 805
 - MakeNewSymScaledMatrix, 805
 - operator=, 806
 - RowColScaling, 806
 - scaling_, 806
 - SymScaledMatrixSpace, 805
 - unscaled_matrix_space_, 806
 - UnscaledMatrixSpace, 806
- Ipopt::SymTMatrix, 806
 - ~SymTMatrix, 809
 - ComputeRowAMaxImpl, 810
 - FillStruct, 809
 - FillValues, 810
 - IsValidNumbersImpl, 810
 - initialized_, 810
 - Irows, 809
 - Jcols, 809
 - MultVectorImpl, 810
 - Nonzeros, 809
 - operator=, 810
 - owner_space_, 810
 - PrintImpl, 810
 - SetValues, 809
 - SymTMatrix, 809
 - Values, 809
 - values_, 810
- Ipopt::SymTMatrixSpace, 811
 - ~SymTMatrixSpace, 812
 - AllocateInternalStorage, 813
 - FreeInternalStorage, 813
 - iRows_, 813
 - Irows, 813
 - jCols_, 813
 - Jcols, 813
 - MakeNewSymMatrix, 812
 - MakeNewSymTMatrix, 812
 - nonZeros_, 813
 - Nonzeros, 812
 - SymTMatrix, 813
 - SymTMatrixSpace, 812
- Ipopt::TDependencyDetector, 817
 - ~TDependencyDetector, 818
 - DetermineDependentRows, 818
 - InitializImpl, 818
 - operator=, 818
 - TDependencyDetector, 818
- Ipopt::TNLP, 829
 - ~TNLP, 832

- eval_f, [834](#)
- eval_g, [834](#)
- eval_grad_f, [834](#)
- eval_h, [834](#)
- eval_jac_g, [834](#)
- finalize_metadata, [835](#)
- finalize_solution, [835](#)
- get_bounds_info, [833](#)
- get_constraints_linearity, [833](#)
- get_list_of_nonlinear_variables, [835](#)
- get_nlp_info, [832](#)
- get_number_of_nonlinear_variables, [835](#)
- get_scaling_parameters, [833](#)
- get_starting_point, [833](#)
- get_var_con_metadata, [832](#)
- get_variables_linearity, [833](#)
- get_warm_start_iterate, [834](#)
- IndexStyleEnum, [832](#)
- IntegerMetaDataMapType, [831](#)
- intermediate_callback, [835](#)
- LinearityType, [832](#)
- NumericMetaDataMapType, [831](#)
- operator=, [835](#)
- StringMetaDataMapType, [831](#)
- TNLP, [832](#)
- Ipopt::TNLPAdapter, [836](#)
 - ~TNLPAdapter, [842](#)
 - bound_relax_factor_, [846](#)
 - c_rhs_, [849](#)
 - c_space_, [848](#)
 - CheckDerivatives, [844](#)
 - d_l_space_, [848](#)
 - d_space_, [848](#)
 - d_u_space_, [849](#)
 - dependency_detection_with_rhs_, [847](#)
 - dependency_detector_, [845](#)
 - derivative_test_, [846](#)
 - derivative_test_first_index_, [846](#)
 - derivative_test_perturbation_, [846](#)
 - derivative_test_print_all_, [846](#)
 - derivative_test_tol_, [846](#)
 - DerivativeTestEnum, [841](#)
 - DetermineDependentConstraints, [845](#)
 - Eval_c, [843](#)
 - Eval_d, [843](#)
 - Eval_f, [843](#)
 - Eval_grad_f, [843](#)
 - Eval_h, [843](#)
 - Eval_jac_c, [843](#)
 - Eval_jac_d, [843](#)
 - FinalizeSolution, [844](#)
 - findiff_jac_ia_, [851](#)
 - findiff_jac_ja_, [851](#)
 - findiff_jac_nnz_, [851](#)
 - findiff_jac_postriplet_, [851](#)
 - findiff_perturbation_, [847](#)
 - findiff_x_l_, [851](#)
 - findiff_x_u_, [851](#)
 - fixed_variable_treatment_, [846](#)
 - FixedVariableTreatmentEnum, [841](#)
 - full_g_, [849](#)
 - full_lambda_, [849](#)
 - full_x_, [849](#)
 - GetBoundsInformation, [843](#)
 - GetQuasiNewtonApproximationSpaces, [844](#)
 - GetScalingParameters, [844](#)
 - GetSpaces, [842](#)
 - GetStartingPoint, [843](#)
 - GetWarmStartIterate, [843](#)
 - h_idx_map_, [851](#)
 - Hess_lagrangian_space_, [849](#)
 - hessian_approximation_, [846](#)
 - index_style_, [848](#)
 - initialize_findiff_jac, [845](#)
 - IntermediateCallBack, [844](#)
 - internal_eval_g, [845](#)
 - internal_eval_jac_g, [845](#)
 - Jac_c_space_, [849](#)
 - Jac_d_space_, [849](#)
 - jac_g_, [849](#)
 - jac_idx_map_, [851](#)
 - jacobian_approximation_, [847](#)
 - JacobianApproxEnum, [842](#)
 - jnlst_, [845](#)
 - n_full_g_, [847](#)
 - n_full_x_, [847](#)
 - n_x_fixed_, [848](#)
 - nlp_lower_bound_inf_, [845](#)
 - nlp_upper_bound_inf_, [845](#)
 - num_linear_variables_, [846](#)
 - nz_full_h_, [848](#)
 - nz_full_jac_g_, [847](#)
 - nz_h_, [848](#)
 - nz_jac_c_, [847](#)
 - nz_jac_c_no_extra_, [847](#)
 - nz_jac_d_, [847](#)
 - operator=, [845](#)
 - P_c_g_, [850](#)
 - P_c_g_space_, [850](#)
 - P_d_g_, [851](#)
 - P_d_g_space_, [850](#)
 - P_x_full_x_, [850](#)
 - P_x_full_x_space_, [850](#)
 - P_x_x_L_, [850](#)
 - P_x_x_L_space_, [850](#)
 - P_x_x_U_, [850](#)
 - P_x_x_U_space_, [850](#)
 - pd_l_space_, [849](#)

- pd_u_space_, 849
- point_perturbation_radius_, 847
- ProcessOptions, 842
- px_l_space_, 848
- px_u_space_, 848
- RegisterOptions, 844
- ResortBnds, 845
- ResortG, 845
- ResortX, 845
- TNLPAAdapter, 842
- tnlp, 844
- tnlp_, 845
- tol_, 847
- update_local_lambda, 845
- update_local_x, 845
- warm_start_same_structure_, 846
- x_fixed_map_, 851
- x_l_space_, 848
- x_space_, 848
- x_tag_for_g_, 850
- x_tag_for_iterates_, 849
- x_tag_for_jac_g_, 850
- x_u_space_, 848
- y_c_tag_for_iterates_, 849
- y_d_tag_for_iterates_, 850
- Ipopt::TNLPReducer, 852
 - ~TNLPReducer, 854
 - eval_f, 856
 - eval_g, 856
 - eval_grad_f, 856
 - eval_h, 856
 - eval_jac_g, 856
 - finalize_solution, 856
 - g_keep_map_, 857
 - get_bounds_info, 855
 - get_constraints_linearity, 855
 - get_list_of_nonlinear_variables, 857
 - get_nlp_info, 855
 - get_number_of_nonlinear_variables, 857
 - get_scaling_parameters, 855
 - get_starting_point, 855
 - get_variables_linearity, 855
 - get_warm_start_iterate, 855
 - index_g_skip_, 857
 - index_style_orig_, 857
 - index_x_fix_, 858
 - index_xL_skip_, 858
 - index_xU_skip_, 858
 - intermediate_callback, 856
 - jac_g_skipped_, 858
 - m_orig_, 857
 - m_reduced_, 858
 - n_g_skip_, 857
 - n_x_fix_, 858
 - n_xL_skip_, 858
 - n_xU_skip_, 858
 - nnz_jac_g_orig_, 857
 - nnz_jac_g_reduced_, 858
 - nnz_jac_g_skipped_, 858
 - operator=, 857
 - TNLPReducer, 854
 - tnlp_, 857
- Ipopt::TSymDependencyDetector, 875
 - ~TSymDependencyDetector, 876
 - DetermineDependentRows, 876
 - InitializeImpl, 876
 - jnlst_, 877
 - operator=, 876
 - RegisterOptions, 876
 - TSymDependencyDetector, 876
 - tsym_linear_solver_, 877
- Ipopt::TSymLinearSolver, 877
 - ~TSymLinearSolver, 880
 - airn_, 883
 - ajcn_, 883
 - atag_, 881
 - DetermineDependentRows, 881
 - dim_, 881
 - GiveMatrixToSolver, 881
 - have_structure_, 882
 - IncreaseQuality, 881
 - InitializeImpl, 880
 - InitializeStructure, 881
 - initialized_, 882
 - just_switched_on_scaling_, 882
 - linear_scaling_on_demand_, 882
 - matrix_format_, 883
 - MultiSolve, 880
 - nonzeros_compressed_, 882
 - nonzeros_triplet_, 882
 - NumberOfNegEvals, 880
 - operator=, 881
 - ProvidesDegeneracyDetection, 881
 - ProvidesInertia, 881
 - RegisterOptions, 881
 - scaling_factors_, 882
 - scaling_method_, 882
 - solver_interface_, 882
 - TSymLinearSolver, 880
 - triplet_to_csr_converter_, 883
 - use_scaling_, 882
 - warm_start_same_structure_, 883
- Ipopt::TSymScalingMethod, 883
 - ~TSymScalingMethod, 884
 - ComputeSymTScalingFactors, 885
 - InitializeImpl, 884
 - operator=, 885
 - TSymScalingMethod, 884

- Ipopt::TaggedObject, 813
 - ~TaggedObject, 816
 - cache_priority_, 817
 - GetTag, 816
 - HasChanged, 816
 - ObjectChanged, 816
 - operator=, 816
 - Tag, 816
 - tagcount_, 816
 - TaggedObject, 816
- Ipopt::TimedTask, 819
 - ~TimedTask, 820
 - End, 821
 - end_called_, 822
 - EndIfStarted, 821
 - operator=, 821
 - Reset, 820
 - Start, 820
 - start_called_, 822
 - start_cputime_, 821
 - start_systime_, 821
 - start_walltime_, 822
 - TimedTask, 820
 - total_cputime_, 821
 - total_systime_, 822
 - total_walltime_, 822
 - TotalCpuTime, 821
 - TotalSysTime, 821
 - TotalWallclockTime, 821
- Ipopt::TimingStatistics, 822
 - ~TimingStatistics, 824
 - AcceptTrialPoint, 825
 - AcceptTrialPoint_, 827
 - CheckConvergence, 825
 - CheckConvergence_, 827
 - ComputeAcceptableTrialPoint, 825
 - ComputeAcceptableTrialPoint_, 827
 - ComputeResiduals, 825
 - ComputeResiduals_, 827
 - ComputeSearchDirection, 825
 - ComputeSearchDirection_, 827
 - Initializeliterates, 825
 - Initializeliterates_, 827
 - LinearSystemBackSolve, 826
 - LinearSystemBackSolve_, 828
 - LinearSystemFactorization, 826
 - LinearSystemFactorization_, 828
 - LinearSystemScaling, 825
 - LinearSystemScaling_, 828
 - LinearSystemStructureConverter, 826
 - LinearSystemStructureConverter_, 828
 - LinearSystemStructureConverterInit, 826
 - LinearSystemStructureConverterInit_, 828
 - LinearSystemSymbolicFactorization, 826
 - LinearSystemSymbolicFactorization_, 828
 - operator=, 826
 - OutputIteration, 825
 - OutputIteration_, 827
 - OverallAlgorithm, 824
 - OverallAlgorithm_, 827
 - PDSystemSolverSolveOnce, 825
 - PDSystemSolverSolveOnce_, 827
 - PDSystemSolverTotal, 825
 - PDSystemSolverTotal_, 827
 - PrintAllTimingStatistics, 824
 - PrintProblemStatistics, 825
 - PrintProblemStatistics_, 827
 - QualityFunctionSearch, 826
 - QualityFunctionSearch_, 828
 - ResetTimes, 824
 - StdAugSystemSolverMultiSolve, 825
 - StdAugSystemSolverMultiSolve_, 827
 - Task1, 826
 - Task1_, 828
 - Task2, 826
 - Task2_, 828
 - Task3, 826
 - Task3_, 828
 - Task4, 826
 - Task4_, 828
 - Task5, 826
 - Task5_, 828
 - Task6, 826
 - Task6_, 828
 - TimingStatistics, 824
 - TryCorrector, 826
 - TryCorrector_, 828
 - UpdateBarrierParameter, 825
 - UpdateBarrierParameter_, 827
 - UpdateHessian, 825
 - UpdateHessian_, 827
- Ipopt::TransposeMatrix, 859
 - ~TransposeMatrix, 860
 - ComputeColAMaxImpl, 861
 - ComputeRowAMaxImpl, 861
 - HasValidNumbersImpl, 861
 - MultVectorImpl, 860
 - operator=, 861
 - orig_matrix_, 862
 - OrigMatrix, 860
 - PrintImpl, 861
 - TransMultVectorImpl, 861
 - TransposeMatrix, 860
- Ipopt::TransposeMatrixSpace, 862
 - ~TransposeMatrixSpace, 863
 - MakeNew, 863
 - MakeNewOrigMatrix, 863
 - MakeNewTransposeMatrix, 863

- operator=, 864
- orig_matrix_space_, 864
- TransposeMatrixSpace, 863
- Ipopt::TripletHelper, 866
 - FillRowCol, 868
 - FillRowCol_, 868–870
 - FillValues, 868
 - FillValues_, 869, 870
 - FillValuesFromVector, 868
 - GetNumberEntries, 868
 - GetNumberEntries_, 868
 - PutValuesInVector, 868
- Ipopt::TripletToCSRConverter, 870
 - ~TripletToCSRConverter, 872
 - ConvertValues, 873
 - dim_, 874
 - ETriFull, 872
 - hf_, 873
 - IA, 873
 - iPosFirst, 873
 - ia_, 873
 - InitializeConverter, 873
 - initialized_, 874
 - ipos_double_compressed_, 874
 - ipos_double_triplet_, 874
 - ipos_first_, 874
 - JA, 873
 - ja_, 874
 - nonzeros_compressed_, 874
 - nonzeros_triplet_, 874
 - num_doubles_, 874
 - offset_, 873
 - operator=, 873
 - TripletToCSRConverter, 872
- Ipopt::TripletToCSRConverter::TripletEntry, 864
 - ~TripletEntry, 865
 - i_pos_triplet_, 866
 - i_row_, 866
 - IRow, 865
 - j_col_, 866
 - JCol, 865
 - operator<, 866
 - operator=, 865
 - PosTriplet, 866
 - Set, 865
 - TripletEntry, 865
- Ipopt::UserScaling, 885
 - ~UserScaling, 886
 - DetermineScalingParametersImpl, 886
 - nlp_, 887
 - operator=, 887
 - UserScaling, 886
- Ipopt::Vector, 887
 - ~Vector, 891
- AddOneVector, 894
- AddScalar, 893
- AddScalarImpl, 896
- AddTwoVectors, 894
- AddTwoVectorsImpl, 897
- AddVectorQuotient, 894
- AddVectorQuotientImpl, 897
- Amax, 892
- amax_cache_tag_, 898
- AmaxImpl, 895
- Asum, 892
- asum_cache_tag_, 898
- AsumImpl, 895
- Axpy, 892
- AxpyImpl, 895
- cached_amax_, 898
- cached_asum_, 898
- cached_max_, 898
- cached_min_, 898
- cached_nrm2_, 898
- cached_sum_, 899
- cached_sumlogs_, 899
- cached_valid_, 899
- Copy, 892
- CopyImpl, 895
- Dim, 894
- Dot, 892
- dot_cache_, 898
- DotImpl, 895
- ElementWiseAbs, 893
- ElementWiseAbsImpl, 896
- ElementWiseDivide, 893
- ElementWiseDivideImpl, 896
- ElementWiseMax, 893
- ElementWiseMaxImpl, 896
- ElementWiseMin, 893
- ElementWiseMinImpl, 896
- ElementWiseMultiply, 893
- ElementWiseMultiplyImpl, 896
- ElementWiseReciprocal, 893
- ElementWiseReciprocalImpl, 896
- ElementWiseSgn, 893
- ElementWiseSgnImpl, 896
- ElementWiseSqrt, 893
- ElementWiseSqrtImpl, 896
- FracToBound, 894
- FracToBoundImpl, 897
- HasValidNumbers, 894
- HasValidNumbersImpl, 897
- MakeNew, 892
- MakeNewCopy, 892
- Max, 893
- max_cache_tag_, 898
- MaxImpl, 896

- Min, 894
- min_cache_tag_, 898
- MinImpl, 897
- Nrm2, 892
- nrm2_cache_tag_, 898
- Nrm2Impl, 895
- operator=, 897
- owner_space_, 898
- OwnerSpace, 895
- Print, 895
- PrintImpl, 897
- Scal, 892
- ScallImpl, 895
- Set, 892
- SetImpl, 895
- Sum, 894
- sum_cache_tag_, 898
- SumImpl, 897
- SumLogs, 894
- SumLogsImpl, 897
- sumlogs_cache_tag_, 899
- valid_cache_tag_, 899
- Vector, 891
- Ipopt::VectorSpace, 899
 - ~VectorSpace, 900
 - Dim, 901
 - dim_, 901
 - MakeNew, 901
 - operator=, 901
 - VectorSpace, 900
- Ipopt::WarmStartIterateInitializer, 901
 - ~WarmStartIterateInitializer, 903
 - adapt_to_target_mu, 904
 - InitializeImpl, 903
 - operator=, 903
 - process_target_mu, 903
 - RegisterOptions, 903
 - SetInitialIterates, 903
 - warm_start_bound_frac_, 904
 - warm_start_bound_push_, 904
 - warm_start_entire_iterate_, 904
 - warm_start_mult_bound_push_, 904
 - warm_start_mult_init_max_, 904
 - warm_start_slack_bound_frac_, 904
 - warm_start_slack_bound_push_, 904
 - warm_start_target_mu_, 904
 - WarmStartIterateInitializer, 903
- Ipopt::WsmpSolverInterface, 905
 - ~WsmpSolverInterface, 908
 - a_, 910
 - DPARM_, 911
 - DetermineDependentRows, 909
 - dim_, 909
 - Factorization, 909
 - factorizations_since_recomputed_ordering_, 911
 - GetValuesArrayPtr, 908
 - have_symbolic_factorization_, 911
 - INVP_, 912
 - IPARM_, 911
 - IncreaseQuality, 908
 - InitializeImpl, 908
 - InitializeStructure, 908
 - initialized_, 911
 - InternalSymFact, 909
 - MRP_, 912
 - matrix_file_number_, 911
 - MatrixFormat, 909
 - MultiSolve, 908
 - negevals_, 911
 - nonzeros_, 910
 - NumberOfNegEvals, 908
 - operator=, 909
 - PERM_, 911
 - pivotol_changed_, 911
 - printed_num_threads_, 911
 - ProvidesDegeneracyDetection, 909
 - ProvidesInertia, 908
 - RegisterOptions, 909
 - skip_inertia_check_, 910
 - Solve, 909
 - SymbolicFactorization, 909
 - wsmp_no_pivoting_, 910
 - wsmp_num_threads_, 910
 - wsmp_pivotol_, 910
 - wsmp_pivotolmax_, 910
 - wsmp_scaling_, 910
 - wsmp_singularity_threshold_, 910
 - wsmp_write_matrix_iteration_, 910
 - WsmpSolverInterface, 908
- Ipopt::ZeroMatrix, 912
 - ~ZeroMatrix, 913
 - ComputeColAMaxImpl, 914
 - ComputeRowAMaxImpl, 914
 - MultVectorImpl, 914
 - operator=, 914
 - PrintImpl, 914
 - TransMultVectorImpl, 914
 - ZeroMatrix, 913
- Ipopt::ZeroMatrixSpace, 914
 - ~ZeroMatrixSpace, 915
 - MakeNew, 916
 - MakeNewZeroMatrix, 916
 - operator=, 916
 - ZeroMatrixSpace, 915, 916
- Ipopt::ZeroSymMatrix, 916
 - ~ZeroSymMatrix, 918
 - ComputeColAMaxImpl, 918
 - ComputeRowAMaxImpl, 918

- MultVectorImpl, 918
- operator=, 919
- PrintImpl, 919
- TransMultVectorImpl, 918
- ZeroSymMatrix, 918
- Ipopt::ZeroSymMatrixSpace, 919
 - ~ZeroSymMatrixSpace, 920
 - MakeNew, 920
 - MakeNewSymMatrix, 920
 - MakeNewZeroSymMatrix, 921
 - operator=, 921
 - ZeroSymMatrixSpace, 920
- ipopt_dbg_smartptr_verbosity
 - IpSmartPtr.hpp, 974
- ipopt_name_
 - Ipopt::AmplOptionsList::PrivatInfo, 661
- ipopt_option_name_
 - Ipopt::AmplOptionsList::AmplOption, 71
- IpoptAdditionalCq
 - Ipopt::IpoptAdditionalCq, 330
- IpoptAdditionalData
 - Ipopt::IpoptAdditionalData, 331
- IpoptAlgorithm
 - Ipopt::IpoptAlgorithm, 335
- IpoptApplication
 - Ipopt::IpoptApplication, 341
- IpoptCQObject
 - Ipopt::IpoptApplication, 343
- IpoptCalculatedQuantities
 - Ipopt::IpoptCalculatedQuantities, 355
- IpoptData
 - Ipopt::IpoptData, 379
- IpoptDataObject
 - Ipopt::IpoptApplication, 343
- IpoptException
 - Ipopt::IpoptException, 390
- IpoptNLP
 - Ipopt::IpoptNLP, 394
- IpoptNLPObject
 - Ipopt::IpoptApplication, 343
- IpoptName
 - Ipopt::AmplOptionsList::PrivatInfo, 661
- IpoptOptionName
 - Ipopt::AmplOptionsList::AmplOption, 71
- IpoptProblem
 - IpStdCInterface.h, 1002
- ipos_double_compressed_
 - Ipopt::TripletToCSRConverter, 874
- ipos_double_triplet_
 - Ipopt::TripletToCSRConverter, 874
- ipos_first_
 - Ipopt::TripletToCSRConverter, 874
- Irows
 - Ipopt::GenTMatrix, 267
- Ipopt::GenTMatrixSpace, 271
- Ipopt::SymTMatrix, 809
- Ipopt::SymTMatrixSpace, 813
- is_pardiso_
 - Ipopt::InexactPDSolver, 317
- IsAcceptableToCurrentFilter
 - Ipopt::FilterLSAcceptor, 252
- IsAcceptableToCurrentIterate
 - Ipopt::FilterLSAcceptor, 252
 - Ipopt::InexactLSAcceptor, 302
 - Ipopt::PenaltyLSAcceptor, 652
- IsAcceptableToPiecewisePenalty
 - Ipopt::CGPenaltyLSAcceptor, 134
- IsAccepted
 - Ipopt::Journal, 439
- IsCompConst
 - Ipopt::CompoundVector, 172
- IsCompNull
 - Ipopt::CompoundVector, 172
- IsFiniteNumber
 - Ipopt, 52
- IsFtype
 - Ipopt::FilterLSAcceptor, 252
- IsHomogeneous
 - Ipopt::DenseVector, 205
- IsMatrixFromSpace
 - Ipopt::MatrixSpace, 541
- IsNull
 - Ipopt, 50
 - Ipopt::SmartPtr, 741
- IsPiecewisePenaltyListEmpty
 - Ipopt::PiecewisePenalty, 657
- IsSquareProblem
 - Ipopt::IpoptCalculatedQuantities, 363
- IsStale
 - Ipopt::DependentResult, 215
- IsValid
 - Ipopt, 50
 - Ipopt::SmartPtr, 741
- IsValidIntegerSetting
 - Ipopt::RegisteredOption, 685
- IsValidNumberSetting
 - Ipopt::RegisteredOption, 684
- IsValidStringSetting
 - Ipopt::RegisteredOption, 685
- ispare
 - ma77_control_d, 503
 - ma77_info_d, 506
 - ma97_control_d, 523
 - ma97_info, 525
- iter
 - Ipopt::FilterEntry, 246
- iter_
 - Ipopt::FilterEntry, 246

- iter_count
 - Ipopt::IpoptData, 382
- iter_count_
 - Ipopt::IpoptData, 386
- iter_output_
 - Ipopt::IpoptAlgorithm, 337
- iterate_initializer_
 - Ipopt::IpoptAlgorithm, 337
- IterateInitializer
 - Ipopt::IterateInitializer, 400
- iterates_space_
 - Ipopt::IpoptData, 388
- IteratesVector
 - Ipopt::IteratesVector, 404
- IteratesVectorSpace
 - Ipopt::IteratesVectorSpace, 413
- IterationCount
 - Ipopt::SolveStatistics, 744
- IterationOutput
 - Ipopt::IterationOutput, 416, 417
- IterativePardisoSolverInterface
 - Ipopt::IterativePardisoSolverInterface, 421
- IterativeSolverTerminationTester
 - Ipopt::IterativeSolverTerminationTester, 429
- IterativeWsmvSolverInterface
 - Ipopt::IterativeWsmvSolverInterface, 433
- iw_
 - Ipopt::Ma27TSolverInterface, 490
- J
- J1_
 - Ipopt::LowRankAugSystemSolver, 469
- J2_
 - Ipopt::LowRankAugSystemSolver, 469
- J_ALL
 - Ipopt, 46
- J_BARRIER_UPDATE
 - Ipopt, 47
- J_DBG
 - Ipopt, 47
- J_DETAILED
 - Ipopt, 46
- J_DOCUMENTATION
 - Ipopt, 47
- J_ERROR
 - Ipopt, 46
- J_FRAC_TO_BOUND
 - Ipopt, 47
- J_HESSIAN_APPROXIMATION
 - Ipopt, 47
- J_INITIALIZATION
 - Ipopt, 47
- J_INSUPPRESSIBLE
 - Ipopt, 46
- J_ITERSUMMARY
 - Ipopt, 46
- J_LAST_CATEGORY
 - Ipopt, 47
- J_LAST_LEVEL
 - Ipopt, 46
- J_LINE_SEARCH
 - Ipopt, 47
- J_LINEAR_ALGEBRA
 - Ipopt, 47
- J_MAIN
 - Ipopt, 47
- J_MATRIX
 - Ipopt, 46
- J_MOREDETAILED
 - Ipopt, 46
- J_MOREMATRIX
 - Ipopt, 46
- J_MOREVECTOR
 - Ipopt, 46
- J_NLP
 - Ipopt, 47
- J_NONE
 - Ipopt, 46
- J_SOLUTION
 - Ipopt, 47
- J_SOLVE_PD_SYSTEM
 - Ipopt, 47
- J_STATISTICS
 - Ipopt, 47
- J_STRONGWARNING
 - Ipopt, 46
- J_SUMMARY
 - Ipopt, 46
- J_TIMING_STATISTICS
 - Ipopt, 47
- J_USER1
 - Ipopt, 47
- J_USER10
 - Ipopt, 47
- J_USER11
 - Ipopt, 47
- J_USER12
 - Ipopt, 47
- J_USER13
 - Ipopt, 47
- J_USER14
 - Ipopt, 47
- J_USER15
 - Ipopt, 47
- J_USER16
 - Ipopt, 47
- J_USER17
 - Ipopt, 47

J_USER2
 Ipopt, [47](#)
 J_USER3
 Ipopt, [47](#)
 J_USER4
 Ipopt, [47](#)
 J_USER5
 Ipopt, [47](#)
 J_USER6
 Ipopt, [47](#)
 J_USER7
 Ipopt, [47](#)
 J_USER8
 Ipopt, [47](#)
 J_USER9
 Ipopt, [47](#)
 J_USER_APPLICATION
 Ipopt, [47](#)
 J_VECTOR
 Ipopt, [46](#)
 J_WARNING
 Ipopt, [46](#)
 JAC_EXACT
 Ipopt::TNLPAdapter, [842](#)
 JAC_FINDIFF_VALUES
 Ipopt::TNLPAdapter, [842](#)
 J_c_ext
 Ipopt::LowRankSSAugSystemSolver, [476](#)
 j_c_tag_
 Ipopt::GenAugSystemSolver, [260](#)
 Ipopt::LowRankAugSystemSolver, [468](#)
 Ipopt::LowRankSSAugSystemSolver, [475](#)
 Ipopt::StdAugSystemSolver, [765](#)
 j_col_
 Ipopt::TripletToCSRConverter::TripletEntry, [866](#)
 j_d_tag_
 Ipopt::GenAugSystemSolver, [261](#)
 Ipopt::LowRankAugSystemSolver, [468](#)
 Ipopt::LowRankSSAugSystemSolver, [475](#)
 Ipopt::StdAugSystemSolver, [765](#)
 JA
 Ipopt::TripletToCSRConverter, [873](#)
 JCol
 Ipopt::TripletToCSRConverter::TripletEntry, [865](#)
 jCols_
 Ipopt::GenTMatrixSpace, [271](#)
 Ipopt::SymTMatrixSpace, [813](#)
 ja_
 Ipopt::TripletToCSRConverter, [874](#)
 jac_c
 Ipopt::IpoptNLP, [395](#)
 Ipopt::OrigIpoptNLP, [611](#)
 Ipopt::RestIpoptNLP, [705](#)
 jac_c_cache_
 Ipopt::OrigIpoptNLP, [617](#)
 Ipopt::OrigIpoptNLP, [619](#)
 Ipopt::OrigIpoptNLP, [615](#)
 Ipopt::OrigIpoptNLP, [620](#)
 Ipopt::IpoptNLP, [397](#)
 Ipopt::OrigIpoptNLP, [614](#)
 Ipopt::RestIpoptNLP, [708](#)
 Ipopt::OrigIpoptNLP, [620](#)
 Ipopt::RestIpoptNLP, [712](#)
 Jac_c_space_
 Ipopt::TNLPAdapter, [849](#)
 jac_c_space_
 Ipopt::OrigIpoptNLP, [616](#)
 Ipopt::RestIpoptNLP, [710](#)
 jac_d
 Ipopt::IpoptNLP, [395](#)
 Ipopt::OrigIpoptNLP, [611](#)
 Ipopt::RestIpoptNLP, [705](#)
 jac_d_cache_
 Ipopt::OrigIpoptNLP, [617](#)
 jac_d_constant_
 Ipopt::OrigIpoptNLP, [619](#)
 jac_d_eval_time
 Ipopt::OrigIpoptNLP, [615](#)
 jac_d_eval_time_
 Ipopt::OrigIpoptNLP, [620](#)
 jac_d_evals
 Ipopt::IpoptNLP, [397](#)
 Ipopt::OrigIpoptNLP, [614](#)
 Ipopt::RestIpoptNLP, [708](#)
 jac_d_evals_
 Ipopt::OrigIpoptNLP, [620](#)
 Ipopt::RestIpoptNLP, [712](#)
 Jac_d_space_
 Ipopt::TNLPAdapter, [849](#)
 jac_d_space_
 Ipopt::OrigIpoptNLP, [617](#)
 Ipopt::RestIpoptNLP, [710](#)
 jac_degenerate_
 Ipopt::CGPerturbationHandler, [145](#)
 Ipopt::PDPerturbationHandler, [641](#)
 jac_g_
 Ipopt::TNLPAdapter, [849](#)
 jac_g_skipped_
 Ipopt::TNLPReducer, [858](#)
 jac_idx_map_
 Ipopt::TNLPAdapter, [851](#)
 jacobian_approximation_
 Ipopt::TNLPAdapter, [847](#)

- JacobianApproxEnum
 - Ipopt::TNLPAdapter, [842](#)
- Jcols
 - Ipopt::GenTMatrix, [267](#)
 - Ipopt::GenTMatrixSpace, [271](#)
 - Ipopt::SymTMatrix, [809](#)
 - Ipopt::SymTMatrixSpace, [813](#)
- Jnlst
 - Ipopt::AlgorithmStrategyObject, [69](#)
 - Ipopt::AmplOptionsList::PrivatInfo, [661](#)
 - Ipopt::IpoptApplication, [342](#)
 - Ipopt::NLPScalingObject, [587](#)
- jnlst_
 - Ipopt::AlgorithmStrategyObject, [69](#)
 - Ipopt::AmplOptionsList::PrivatInfo, [662](#)
 - Ipopt::AmplTNLP, [85](#)
 - Ipopt::IpoptApplication, [344](#)
 - Ipopt::Ma28TDependencyDetector, [493](#)
 - Ipopt::NLPScalingObject, [588](#)
 - Ipopt::OptionsList, [602](#)
 - Ipopt::OrigIpoptNLP, [616](#)
 - Ipopt::StdInterfaceTNLP, [772](#)
 - Ipopt::TNLPAdapter, [845](#)
 - Ipopt::TSymDependencyDetector, [877](#)
- Journal
 - Ipopt::Journal, [438](#)
- Journalist
 - Ipopt::Journalist, [442](#)
- journals_
 - Ipopt::Journalist, [443](#)
- jump_for_tiny_step_
 - Ipopt::CGPenaltyLSAceptor, [138](#)
- just_switched_on_scaling_
 - Ipopt::TSymLinearSolver, [882](#)
- K
- KKT_ERROR
 - Ipopt::AdaptiveMuUpdate, [58](#)
- KKTPenaltyInitialized
 - Ipopt::CGPenaltyData, [127](#)
- kappa_d_
 - Ipopt::IpoptCalculatedQuantities, [366](#)
- kappa_resto_
 - Ipopt::RestoConvergenceCheck, [696](#)
- kappa_sigma_
 - Ipopt::IpoptAlgorithm, [337](#)
- kappa_soc_
 - Ipopt::CGPenaltyLSAceptor, [137](#)
 - Ipopt::FilterLSAceptor, [254](#)
 - Ipopt::PenaltyLSAceptor, [653](#)
- kappa_x_dis_
 - Ipopt::CGSearchDirCalculator, [150](#)
- kappa_y_dis_
 - Ipopt::CGSearchDirCalculator, [150](#)
- keep_
 - Ipopt::Ma77SolverInterface, [511](#)
 - Ipopt::Ma86SolverInterface, [520](#)
- keywds_
 - Ipopt::AmplOptionsList, [74](#)
- keyword
 - IpStdCInterface.h, [1005](#)
- Keywords
 - Ipopt::AmplOptionsList, [74](#)
- kkt_error_
 - Ipopt::SolveStatistics, [747](#)
- kkt_penalty_initialized_
 - Ipopt::CGPenaltyData, [128](#)
- L
- LIMITED_MEMORY
 - Ipopt, [45](#)
- LINEAR
 - Ipopt::TNLP, [832](#)
- LOCAL_INFEASIBILITY
 - Ipopt, [48](#)
- LSACCEPTOR_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, [100](#)
- LU
 - Ipopt::DenseGenMatrix, [190](#)
- L_
 - Ipopt::LimMemQuasiNewtonUpdater, [456](#)
- L_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [458](#)
- l_workspace
 - mc68_info, [546](#)
- LMInitialization
 - Ipopt::LimMemQuasiNewtonUpdater, [450](#)
- LMUpdateType
 - Ipopt::LimMemQuasiNewtonUpdater, [450](#)
- LRMultVector
 - Ipopt::MultiVectorMatrix, [556](#)
- LSL_HSLLibraryName
 - HSLLoader.h, [991](#)
- LSL_PardisoLibraryName
 - PardisoLoader.h, [994](#)
- LSL_isHSLLoaded
 - HSLLoader.h, [990](#)
- LSL_isMA27available
 - HSLLoader.h, [990](#)
- LSL_isMA28available
 - HSLLoader.h, [990](#)
- LSL_isMA57available
 - HSLLoader.h, [990](#)
- LSL_isMA77available
 - HSLLoader.h, [990](#)
- LSL_isMA86available
 - HSLLoader.h, [991](#)
- LSL_isMA97available

- HSLLoader.h, 991
- LSL_isMC19available
 - HSLLoader.h, 991
- LSL_isMC68available
 - HSLLoader.h, 991
- LSL_isPardisoLoaded
 - PardisoLoader.h, 994
- LSL_loadHSL
 - HSLLoader.h, 989
- LSL_loadLib
 - LibraryHandler.h, 993
- LSL_loadPardisoLib
 - PardisoLoader.h, 993
- LSL_setMA27
 - HSLLoader.h, 991
- LSL_setMA28
 - HSLLoader.h, 991
- LSL_setMA57
 - HSLLoader.h, 991
- LSL_setMA77
 - HSLLoader.h, 991
- LSL_setMA86
 - HSLLoader.h, 992
- LSL_setMA97
 - HSLLoader.h, 992
- LSL_setMC19
 - HSLLoader.h, 992
- LSL_setMC68
 - HSLLoader.h, 992
- LSL_unloadHSL
 - HSLLoader.h, 990
- LSL_unloadLib
 - LibraryHandler.h, 993
- LSL_unloadPardisoLib
 - PardisoLoader.h, 994
- LUSolveMatrix
 - Ipopt::DenseGenMatrix, 192
- LUSolveVector
 - Ipopt::DenseGenMatrix, 192
- la_
 - Ipopt::Ma27TSolverInterface, 491
- la_increase_
 - Ipopt::Ma27TSolverInterface, 491
- la_init_factor_
 - Ipopt::Ma27TSolverInterface, 490
- lambda_sol_
 - Ipopt::AmplTNLP, 86
 - Ipopt::StdInterfaceTNLP, 775
- last_Av_norm_
 - Ipopt::InexactPDTerminationTester, 323
- last_eta_
 - Ipopt::LimMemQuasiNewtonUpdater, 455
- last_grad_f_
 - Ipopt::LimMemQuasiNewtonUpdater, 457
- last_info_ls_count_
 - Ipopt::InexactPDSolver, 317
- last_iter_
 - Ipopt::InexactNormalTerminationTester, 313
 - Ipopt::InexactPDTerminationTester, 323
- last_jac_c_
 - Ipopt::LimMemQuasiNewtonUpdater, 457
- last_jac_d_
 - Ipopt::LimMemQuasiNewtonUpdater, 457
- last_mu_
 - Ipopt::BacktrackingLineSearch, 105
- last_nu_
 - Ipopt::InexactLSAcceptor, 304
 - Ipopt::PenaltyLSAcceptor, 654
- last_nu_low_
 - Ipopt::InexactLSAcceptor, 304
- last_obj_val_
 - Ipopt::OptimalityErrorConvergenceCheck, 597
- last_obj_val_iter_
 - Ipopt::OptimalityErrorConvergenceCheck, 597
- last_rejection_due_to_filter_
 - Ipopt::FilterLSAcceptor, 255
- last_tr_inactive_
 - Ipopt::InexactDoglegNormalStep, 296
- last_tt1_norm_
 - Ipopt::InexactPDTerminationTester, 323
- last_x_
 - Ipopt::LimMemQuasiNewtonUpdater, 457
- least_square_init_duals_
 - Ipopt::DefaultIterateInitializer, 187
- least_square_init_primal_
 - Ipopt::DefaultIterateInitializer, 187
- least_square_mults
 - Ipopt::DefaultIterateInitializer, 186
- LeastSquareMultipliers
 - Ipopt::LeastSquareMultipliers, 445
- LibraryHandler.h
 - LSL_loadLib, 993
 - LSL_unloadLib, 993
 - soHandle_t, 992
- LimMemQuasiNewtonUpdater
 - Ipopt::LimMemQuasiNewtonUpdater, 451
- limited_memory_init_val_
 - Ipopt::LimMemQuasiNewtonUpdater, 454
- limited_memory_initialization_
 - Ipopt::LimMemQuasiNewtonUpdater, 454
- limited_memory_max_history_
 - Ipopt::LimMemQuasiNewtonUpdater, 454
- limited_memory_max_skipping_
 - Ipopt::LimMemQuasiNewtonUpdater, 454
- limited_memory_special_for_resto_
 - Ipopt::LimMemQuasiNewtonUpdater, 455
- limited_memory_update_type_
 - Ipopt::LimMemQuasiNewtonUpdater, 454

- LinAlg/lpBlas.hpp, 1008
- LinAlg/lpCompoundMatrix.hpp, 1009
- LinAlg/lpCompoundSymMatrix.hpp, 1009
- LinAlg/lpCompoundVector.hpp, 1010
- LinAlg/lpDenseGenMatrix.hpp, 1010
- LinAlg/lpDenseSymMatrix.hpp, 1011
- LinAlg/lpDenseVector.hpp, 1011
- LinAlg/lpDiagMatrix.hpp, 1012
- LinAlg/lpExpandedMultiVectorMatrix.hpp, 1012
- LinAlg/lpExpansionMatrix.hpp, 1012
- LinAlg/lpIdentityMatrix.hpp, 1013
- LinAlg/lpLapack.hpp, 1013
- LinAlg/lpLowRankUpdateSymMatrix.hpp, 1014
- LinAlg/lpMatrix.hpp, 1014
- LinAlg/lpMultiVectorMatrix.hpp, 1015
- LinAlg/lpScaledMatrix.hpp, 1015
- LinAlg/lpSumMatrix.hpp, 1015
- LinAlg/lpSumSymMatrix.hpp, 1016
- LinAlg/lpSymMatrix.hpp, 1016
- LinAlg/lpSymScaledMatrix.hpp, 1017
- LinAlg/lpTransposeMatrix.hpp, 1017
- LinAlg/lpVector.hpp, 1017
- LinAlg/lpZeroMatrix.hpp, 1018
- LinAlg/lpZeroSymMatrix.hpp, 1018
- LinAlg/TMatrices/lpGenTMatrix.hpp, 1019
- LinAlg/TMatrices/lpSymTMatrix.hpp, 1019
- LinAlg/TMatrices/lpTripletHelper.hpp, 1020
- line_number_
 - Ipopt::IpoptException, 391
- line_search_
 - Ipopt::IpoptAlgorithm, 337
- LineSearch
 - Ipopt::LineSearch, 460
- linear_scaling_on_demand_
 - Ipopt::TSymLinearSolver, 882
- linear_solver_
 - Ipopt::IpoptAlgorithm, 338
- LinearSystemBackSolve
 - Ipopt::TimingStatistics, 826
- LinearSystemBackSolve_
 - Ipopt::TimingStatistics, 828
- LinearSystemFactorization
 - Ipopt::TimingStatistics, 826
- LinearSystemFactorization_
 - Ipopt::TimingStatistics, 828
- LinearSystemScaling
 - Ipopt::TimingStatistics, 825
- LinearSystemScaling_
 - Ipopt::TimingStatistics, 828
- LinearSystemStructureConverter
 - Ipopt::TimingStatistics, 826
- LinearSystemStructureConverter_
 - Ipopt::TimingStatistics, 828
- LinearSystemStructureConverterInit
 - Ipopt::TimingStatistics, 826
- LinearSystemStructureConverterInit_
 - Ipopt::TimingStatistics, 828
- LinearSystemSymbolicFactorization
 - Ipopt::TimingStatistics, 826
- LinearSystemSymbolicFactorization_
 - Ipopt::TimingStatistics, 828
- LinearityType
 - Ipopt::TNLP, 832
- linesearch_
 - Ipopt::AdaptiveMuUpdate, 62
 - Ipopt::MonotoneMuUpdate, 553
- linsolver_
 - Ipopt::StdAugSystemSolver, 764
- liw_
 - Ipopt::Ma27TSolverInterface, 490
- liw_increase_
 - Ipopt::Ma27TSolverInterface, 491
- liw_init_factor_
 - Ipopt::Ma27TSolverInterface, 489
- lm_skipped_iter_
 - Ipopt::LimMemQuasiNewtonUpdater, 455
- local_inf_Ac_tol_
 - Ipopt::InexactSearchDirCalculator, 326
- long_description_
 - Ipopt::RegisteredOption, 686
- LongDescription
 - Ipopt::RegisteredOption, 681
- LoqoMuOracle
 - Ipopt::LoqoMuOracle, 462, 463
- LowRankAugSystemSolver
 - Ipopt::LowRankAugSystemSolver, 466
- LowRankSSAugSystemSolver
 - Ipopt::LowRankSSAugSystemSolver, 472, 473
- LowRankUpdateSymMatrix
 - Ipopt::LowRankUpdateSymMatrix, 478, 479
- LowRankUpdateSymMatrixSpace
 - Ipopt::LowRankUpdateSymMatrixSpace, 482
- LowRankVectorSpace
 - Ipopt::LowRankUpdateSymMatrix, 479
 - Ipopt::LowRankUpdateSymMatrixSpace, 483
- lower_
 - Ipopt::RegisteredOption, 686
- lower_mu_safeguard
 - Ipopt::AdaptiveMuUpdate, 60
- lower_strict_
 - Ipopt::RegisteredOption, 686
- LowerInteger
 - Ipopt::RegisteredOption, 683
- LowerNumber
 - Ipopt::RegisteredOption, 682
- LowerStrict
 - Ipopt::RegisteredOption, 682
- lowercase

- Ipopt::OptionsList, 601
- lowercase_buffer_
 - Ipopt::OptionsList, 602
- lowrank_vector_space_
 - Ipopt::LowRankUpdateSymMatrixSpace, 483
- lp
 - mc68_control, 544
- ls_counter_
 - Ipopt::CGPenaltyLSAcceptor, 137
- lspare
 - ma77_control_d, 503
 - ma77_info_d, 506
- M
- m
 - IpStdCInterface.h, 1004
- MAKE_CONSTRAINT
 - Ipopt::TNLPAdapter, 841
- MAKE_PARAMETER
 - Ipopt::TNLPAdapter, 841
- MAX_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, 100
- MAXITER_EXCEEDED
 - Ipopt, 48
 - Ipopt::ConvergenceCheck, 181
- MIN_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, 100
- MIN_DUAL_INFEAS_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, 100
- MODIFY_HESSIAN
 - Ipopt::IterativeSolverTerminationTester, 429
- m_orig_
 - Ipopt::TNLPReducer, 857
- m_reduced_
 - Ipopt::TNLPReducer, 858
- MAXFCT_
 - Ipopt::IterativePardisoSolverInterface, 426
 - Ipopt::PardisoSolverInterface, 630
- MNUM_
 - Ipopt::IterativePardisoSolverInterface, 426
 - Ipopt::PardisoSolverInterface, 630
- MRP_
 - Ipopt::WsmvSolverInterface, 912
- MSGVLV_
 - Ipopt::IterativePardisoSolverInterface, 426
 - Ipopt::PardisoSolverInterface, 630
- MTYPE_
 - Ipopt::IterativePardisoSolverInterface, 426
 - Ipopt::PardisoSolverInterface, 630
- Ma27TSolverInterface
 - Ipopt::Ma27TSolverInterface, 487
- ma27ad_t
 - HSLLoader.h, 986
- ma27bd_t
 - HSLLoader.h, 986
- ma27cd_t
 - HSLLoader.h, 986
- ma27id_t
 - HSLLoader.h, 986
- ma28_pivtol_
 - Ipopt::Ma28TDependencyDetector, 494
- Ma28TDependencyDetector
 - Ipopt::Ma28TDependencyDetector, 493
- ma28ad_t
 - HSLLoader.h, 986
- ma57_pre_alloc_
 - Ipopt::Ma57TSolverInterface, 499
- Ma57TSolverInterface
 - Ipopt::Ma57TSolverInterface, 496
- ma57ad_t
 - HSLLoader.h, 986
- ma57bd_t
 - HSLLoader.h, 986
- ma57cd_t
 - HSLLoader.h, 986
- ma57ed_t
 - HSLLoader.h, 986
- ma57id_t
 - HSLLoader.h, 986
- ma57int
 - IpMa57TSolverInterface.hpp, 960
- ma77_alter
 - hsl_ma77d.h, 950
 - HSLLoader.h, 984
- ma77_alter_d
 - hsl_ma77d.h, 951
- ma77_alter_t
 - HSLLoader.h, 988
- ma77_analyse
 - hsl_ma77d.h, 949, 951
 - HSLLoader.h, 983
- ma77_analyse_t
 - HSLLoader.h, 987
- ma77_control
 - hsl_ma77d.h, 949
 - HSLLoader.h, 983
- ma77_control_d, 500
 - action, 502
 - bits, 501
 - buffer_lpage, 501
 - buffer_npage, 502
 - consist_tol, 503
 - f_arrays, 501
 - file_size, 502
 - inform, 502
 - ispare, 503
 - lspare, 503
 - maxit, 502

- maxstore, [502](#)
- multiplier, [502](#)
- nb54, [502](#)
- nb64, [502](#)
- nbi, [502](#)
- nemin, [502](#)
- print_level, [501](#)
- rspace, [503](#)
- small, [502](#)
- static_, [503](#)
- storage, [502](#)
- storage_indef, [503](#)
- thresh, [502](#)
- u, [503](#)
- umin, [503](#)
- unit_diagnostics, [501](#)
- unit_error, [501](#)
- unit_warning, [501](#)
- ma77_default_control
 - hsl_ma77d.h, [949](#)
 - HSLLoader.h, [983](#)
- ma77_default_control_d
 - hsl_ma77d.h, [950](#)
- ma77_default_control_t
 - HSLLoader.h, [986](#)
- ma77_enquire_indef
 - hsl_ma77d.h, [950](#)
 - HSLLoader.h, [984](#)
- ma77_enquire_indef_d
 - hsl_ma77d.h, [951](#)
- ma77_enquire_indef_t
 - HSLLoader.h, [988](#)
- ma77_enquire_posdef
 - hsl_ma77d.h, [950](#)
 - HSLLoader.h, [984](#)
- ma77_enquire_posdef_d
 - hsl_ma77d.h, [951](#)
- ma77_enquire_posdef_t
 - HSLLoader.h, [987](#)
- ma77_factor
 - hsl_ma77d.h, [949](#)
 - HSLLoader.h, [983](#)
- ma77_factor_d
 - hsl_ma77d.h, [951](#)
- ma77_factor_solve
 - hsl_ma77d.h, [949](#)
 - HSLLoader.h, [983](#)
- ma77_factor_solve_d
 - hsl_ma77d.h, [951](#)
- ma77_factor_solve_t
 - HSLLoader.h, [987](#)
- ma77_factor_t
 - HSLLoader.h, [987](#)
- ma77_finalise
 - hsl_ma77d.h, [950](#)
 - HSLLoader.h, [984](#)
- ma77_finalise_d
 - hsl_ma77d.h, [951](#)
- ma77_finalise_t
 - HSLLoader.h, [988](#)
- ma77_info
 - hsl_ma77d.h, [949](#)
 - HSLLoader.h, [983](#)
- ma77_info_d, [503](#)
 - detlog, [504](#)
 - detsign, [504](#)
 - flag, [504](#)
 - index, [506](#)
 - iostat, [504](#)
 - ispare, [506](#)
 - lspare, [506](#)
 - matrix_dup, [504](#)
 - matrix_outrange, [505](#)
 - matrix_rank, [504](#)
 - maxdepth, [505](#)
 - maxfront, [505](#)
 - minstore, [505](#)
 - ndelay, [505](#)
 - nfactor, [505](#)
 - nflops, [505](#)
 - nio_read, [506](#)
 - nio_write, [506](#)
 - niter, [505](#)
 - nsup, [505](#)
 - ntwo, [505](#)
 - num_file, [506](#)
 - num_neg, [505](#)
 - num_nothresh, [505](#)
 - num_perturbed, [505](#)
 - nwd_read, [506](#)
 - nwd_write, [506](#)
 - rspace, [506](#)
 - stat, [505](#)
 - storage, [506](#)
 - tree_nodes, [506](#)
 - unit_restart, [506](#)
 - unused, [506](#)
 - usmall, [506](#)
- ma77_input_reals
 - hsl_ma77d.h, [949](#)
 - HSLLoader.h, [983](#)
- ma77_input_reals_d
 - hsl_ma77d.h, [951](#)
- ma77_input_reals_t
 - HSLLoader.h, [987](#)
- ma77_input_vars
 - hsl_ma77d.h, [949](#), [950](#)
 - HSLLoader.h, [983](#)

ma77_input_vars_t
HSLLoader.h, [987](#)

ma77_multiply
hsl_ma77d.h, [950](#)

ma77_multiply_d
hsl_ma77d.h, [951](#)

ma77_open
hsl_ma77d.h, [949](#)
HSLLoader.h, [983](#)

ma77_open_d
hsl_ma77d.h, [950](#)

ma77_open_nelt
hsl_ma77d.h, [949](#), [950](#)
HSLLoader.h, [983](#)

ma77_open_nelt_t
HSLLoader.h, [987](#)

ma77_open_t
HSLLoader.h, [987](#)

ma77_resid
hsl_ma77d.h, [950](#)
HSLLoader.h, [983](#)

ma77_resid_d
hsl_ma77d.h, [951](#)

ma77_resid_t
HSLLoader.h, [987](#)

ma77_restart
hsl_ma77d.h, [950](#)
HSLLoader.h, [984](#)

ma77_restart_d
hsl_ma77d.h, [951](#)

ma77_restart_t
HSLLoader.h, [988](#)

ma77_scale
hsl_ma77d.h, [950](#)
HSLLoader.h, [984](#)

ma77_scale_d
hsl_ma77d.h, [951](#)

ma77_scale_t
HSLLoader.h, [987](#)

ma77_solve
hsl_ma77d.h, [949](#)
HSLLoader.h, [983](#)

ma77_solve_d
hsl_ma77d.h, [951](#)

ma77_solve_fredholm
hsl_ma77d.h, [950](#)

ma77_solve_fredholm_d
hsl_ma77d.h, [951](#)

ma77_solve_t
HSLLoader.h, [987](#)

Ma77SolverInterface
Ipopt::Ma77SolverInterface, [509](#)

ma77pkgtype_d
hsl_ma77d.h, [950](#)

HSLLoader.h, [985](#)

ma86_analyse
hsl_ma86d.h, [952](#)
HSLLoader.h, [984](#)

ma86_analyse_d
hsl_ma86d.h, [953](#)

ma86_analyse_t
HSLLoader.h, [988](#)

ma86_control
hsl_ma86d.h, [952](#)
HSLLoader.h, [984](#)

ma86_control_d, [512](#)
action, [513](#)
diagnostics_level, [512](#)
f_arrays, [512](#)
nb, [513](#)
nbi, [513](#)
nemin, [513](#)
pool_size, [513](#)
scaling, [513](#)
small_, [513](#)
static_, [513](#)
u, [513](#)
umin, [513](#)
unit_diagnostics, [513](#)
unit_error, [513](#)
unit_warning, [513](#)

ma86_default_control
hsl_ma86d.h, [952](#)
HSLLoader.h, [984](#)

ma86_default_control_d
hsl_ma86d.h, [953](#)

ma86_default_control_t
HSLLoader.h, [988](#)

ma86_factor
hsl_ma86d.h, [952](#)
HSLLoader.h, [984](#)

ma86_factor_d
hsl_ma86d.h, [953](#)

ma86_factor_solve
hsl_ma86d.h, [952](#)
HSLLoader.h, [984](#)

ma86_factor_solve_d
hsl_ma86d.h, [953](#)

ma86_factor_solve_t
HSLLoader.h, [988](#)

ma86_factor_t
HSLLoader.h, [988](#)

ma86_finalise
hsl_ma86d.h, [953](#)
HSLLoader.h, [984](#)

ma86_finalise_d
hsl_ma86d.h, [953](#)

ma86_finalise_t

- HSLLoader.h, [988](#)
- ma86_info
 - hsl_ma86d.h, [952](#)
 - HSLLoader.h, [984](#)
- ma86_info_d, [514](#)
 - detlog, [514](#)
 - detsign, [514](#)
 - flag, [514](#)
 - matrix_rank, [514](#)
 - maxdepth, [514](#)
 - num_delay, [514](#)
 - num_factor, [515](#)
 - num_flops, [515](#)
 - num_neg, [515](#)
 - num_nodes, [515](#)
 - num_nothresh, [515](#)
 - num_perturbed, [515](#)
 - num_two, [515](#)
 - pool_size, [515](#)
 - stat, [515](#)
 - usmall, [515](#)
- ma86_solve
 - hsl_ma86d.h, [952](#)
 - HSLLoader.h, [984](#)
- ma86_solve_d
 - hsl_ma86d.h, [953](#)
- ma86_solve_t
 - HSLLoader.h, [988](#)
- Ma86SolverInterface
 - Ipopt::Ma86SolverInterface, [518](#)
- ma86pkgtype_d_
 - hsl_ma86d.h, [953](#)
 - HSLLoader.h, [985](#)
- ma86realtype_d_
 - hsl_ma86d.h, [953](#)
 - HSLLoader.h, [985](#)
- ma97_alter
 - hsl_ma97d.h, [955](#)
- ma97_alter_d
 - hsl_ma97d.h, [956](#)
- ma97_analyse
 - hsl_ma97d.h, [955](#)
 - HSLLoader.h, [985](#)
- ma97_analyse_coord
 - hsl_ma97d.h, [955](#)
- ma97_analyse_coord_d
 - hsl_ma97d.h, [956](#)
- ma97_analyse_d
 - hsl_ma97d.h, [956](#)
- ma97_analyse_t
 - HSLLoader.h, [988](#)
- ma97_control
 - hsl_ma97d.h, [954](#)
 - HSLLoader.h, [985](#)
- ma97_control_d, [521](#)
 - action, [521](#)
 - consist_tol, [522](#)
 - f_arrays, [521](#)
 - factor_min, [522](#)
 - ispare, [523](#)
 - multiplier, [522](#)
 - nemin, [521](#)
 - ordering, [522](#)
 - print_level, [522](#)
 - rspare, [523](#)
 - scaling, [522](#)
 - small, [522](#)
 - solve_blas3, [522](#)
 - solve_mf, [522](#)
 - solve_min, [522](#)
 - u, [522](#)
 - unit_diagnostics, [522](#)
 - unit_error, [522](#)
 - unit_warning, [522](#)
- ma97_default_control
 - hsl_ma97d.h, [955](#)
 - HSLLoader.h, [985](#)
- ma97_default_control_d
 - hsl_ma97d.h, [956](#)
- ma97_default_control_t
 - HSLLoader.h, [988](#)
- ma97_enquire_indef
 - hsl_ma97d.h, [955](#)
- ma97_enquire_indef_d
 - hsl_ma97d.h, [956](#)
- ma97_enquire_posdef
 - hsl_ma97d.h, [955](#)
- ma97_enquire_posdef_d
 - hsl_ma97d.h, [956](#)
- ma97_factor
 - hsl_ma97d.h, [955](#)
 - HSLLoader.h, [985](#)
- ma97_factor_d
 - hsl_ma97d.h, [956](#)
- ma97_factor_solve
 - hsl_ma97d.h, [955](#)
 - HSLLoader.h, [985](#)
- ma97_factor_solve_d
 - hsl_ma97d.h, [956](#)
- ma97_factor_solve_t
 - HSLLoader.h, [989](#)
- ma97_factor_t
 - HSLLoader.h, [989](#)
- ma97_finalise
 - hsl_ma97d.h, [955](#)
 - HSLLoader.h, [985](#)
- ma97_finalise_d
 - hsl_ma97d.h, [956](#)

- ma97_finalise_t
 - HSLLoader.h, [989](#)
- ma97_free_akeep
 - hsl_ma97d.h, [955](#)
 - HSLLoader.h, [985](#)
- ma97_free_akeep_d
 - hsl_ma97d.h, [956](#)
- ma97_free_akeep_t
 - HSLLoader.h, [989](#)
- ma97_free_fkeep
 - hsl_ma97d.h, [955](#)
- ma97_free_fkeep_d
 - hsl_ma97d.h, [956](#)
- ma97_info, [523](#)
 - flag, [523](#)
 - flag68, [523](#)
 - flag77, [524](#)
 - hsl_ma97d.h, [954](#)
 - HSLLoader.h, [985](#)
 - ispare, [525](#)
 - matrix_dup, [524](#)
 - matrix_missing_diag, [524](#)
 - matrix_outrange, [524](#)
 - matrix_rank, [524](#)
 - maxdepth, [524](#)
 - maxfront, [524](#)
 - num_delay, [524](#)
 - num_factor, [524](#)
 - num_flops, [524](#)
 - num_neg, [524](#)
 - num_sup, [524](#)
 - num_two, [524](#)
 - ordering, [524](#)
 - rspare, [525](#)
 - stat, [525](#)
- ma97_lmultiply
 - hsl_ma97d.h, [955](#)
- ma97_lmultiply_d
 - hsl_ma97d.h, [957](#)
- ma97_solve
 - hsl_ma97d.h, [955](#)
 - HSLLoader.h, [985](#)
- ma97_solve_d
 - hsl_ma97d.h, [956](#)
- ma97_solve_fredholm
 - hsl_ma97d.h, [955](#)
- ma97_solve_fredholm_d
 - hsl_ma97d.h, [956](#)
- ma97_solve_t
 - HSLLoader.h, [989](#)
- ma97_sparse_fwd_solve
 - hsl_ma97d.h, [956](#)
- ma97_sparse_fwd_solve_d
 - hsl_ma97d.h, [957](#)
- Ma97SolverInterface
 - Ipopt::Ma97SolverInterface, [528](#)
- ma97pkgtype_d_
 - hsl_ma97d.h, [956](#)
 - HSLLoader.h, [985](#)
- ma97realttype_d_
 - hsl_ma97d.h, [956](#)
 - HSLLoader.h, [985](#)
- magic_steps_
 - Ipopt::BacktrackingLineSearch, [104](#)
- MakeNew
 - Ipopt::CompoundMatrixSpace, [159](#)
 - Ipopt::CompoundVectorSpace, [179](#)
 - Ipopt::DenseGenMatrixSpace, [195](#)
 - Ipopt::DenseVectorSpace, [211](#)
 - Ipopt::ExpandedMultiVectorMatrixSpace, [233](#)
 - Ipopt::ExpansionMatrixSpace, [239](#)
 - Ipopt::GenTMatrixSpace, [271](#)
 - Ipopt::IteratesVectorSpace, [414](#)
 - Ipopt::MatrixSpace, [541](#)
 - Ipopt::MultiVectorMatrixSpace, [560](#)
 - Ipopt::ScaledMatrixSpace, [731](#)
 - Ipopt::SumMatrixSpace, [786](#)
 - Ipopt::SymMatrixSpace, [799](#)
 - Ipopt::SymScaledMatrixSpace, [805](#)
 - Ipopt::TransposeMatrixSpace, [863](#)
 - Ipopt::Vector, [892](#)
 - Ipopt::VectorSpace, [901](#)
 - Ipopt::ZeroMatrixSpace, [916](#)
 - Ipopt::ZeroSymMatrixSpace, [920](#)
- MakeNewCompoundMatrix
 - Ipopt::CompoundMatrixSpace, [159](#)
- MakeNewCompoundSymMatrix
 - Ipopt::CompoundSymMatrix, [163](#)
 - Ipopt::CompoundSymMatrixSpace, [167](#)
- MakeNewCompoundVector
 - Ipopt::CompoundVectorSpace, [179](#)
 - Ipopt::IteratesVectorSpace, [414](#)
- MakeNewContainer
 - Ipopt::IteratesVector, [405](#)
- MakeNewCopy
 - Ipopt::Vector, [892](#)
- MakeNewDenseGenMatrix
 - Ipopt::DenseGenMatrix, [191](#)
 - Ipopt::DenseGenMatrixSpace, [195](#)
- MakeNewDenseSymMatrix
 - Ipopt::DenseSymMatrix, [197](#)
 - Ipopt::DenseSymMatrixSpace, [200](#)
- MakeNewDenseVector
 - Ipopt::DenseVector, [204](#)
 - Ipopt::DenseVectorSpace, [211](#)
- MakeNewDiagMatrix
 - Ipopt::DiagMatrixSpace, [221](#)
- MakeNewExpandedMultiVectorMatrix

- Ipopt::ExpandedMultiVectorMatrix, [230](#)
 - Ipopt::ExpandedMultiVectorMatrixSpace, [233](#)
- MakeNewExpansionMatrix
 - Ipopt::ExpansionMatrixSpace, [239](#)
- MakeNewGenTMatrix
 - Ipopt::GenTMatrixSpace, [271](#)
- MakeNewIdentityMatrix
 - Ipopt::IdentityMatrixSpace, [281](#)
- MakeNewIteratesVector
 - Ipopt::IteratesVector, [405](#)
 - Ipopt::IteratesVectorSpace, [413](#), [414](#)
- MakeNewIteratesVectorCopy
 - Ipopt::IteratesVector, [405](#)
- MakeNewLowRankUpdateSymMatrix
 - Ipopt::LowRankUpdateSymMatrixSpace, [483](#)
- MakeNewMultiVectorMatrix
 - Ipopt::MultiVectorMatrix, [556](#)
 - Ipopt::MultiVectorMatrixSpace, [559](#)
- MakeNewOrigMatrix
 - Ipopt::TransposeMatrixSpace, [863](#)
- MakeNewPerturbedPoint
 - Ipopt::PointPerturber, [660](#)
- MakeNewScaledMatrix
 - Ipopt::ScaledMatrixSpace, [731](#)
- MakeNewSumMatrix
 - Ipopt::SumMatrixSpace, [786](#)
- MakeNewSumSymMatrix
 - Ipopt::SumSymMatrixSpace, [792](#)
- MakeNewSymMatrix
 - Ipopt::CompoundSymMatrixSpace, [167](#)
 - Ipopt::DenseSymMatrixSpace, [200](#)
 - Ipopt::DiagMatrixSpace, [221](#)
 - Ipopt::IdentityMatrixSpace, [281](#)
 - Ipopt::LowRankUpdateSymMatrixSpace, [483](#)
 - Ipopt::SumSymMatrixSpace, [792](#)
 - Ipopt::SymMatrixSpace, [799](#)
 - Ipopt::SymScaledMatrixSpace, [805](#)
 - Ipopt::SymTMatrixSpace, [812](#)
 - Ipopt::ZeroSymMatrixSpace, [920](#)
- MakeNewSymScaledMatrix
 - Ipopt::SymScaledMatrixSpace, [805](#)
- MakeNewSymTMatrix
 - Ipopt::SymTMatrixSpace, [812](#)
- MakeNewTransposeMatrix
 - Ipopt::TransposeMatrixSpace, [863](#)
- MakeNewZeroMatrix
 - Ipopt::ZeroMatrixSpace, [916](#)
- MakeNewZeroSymMatrix
 - Ipopt::ZeroSymMatrixSpace, [921](#)
- MakeValidLatexNumber
 - Ipopt::RegisteredOption, [685](#)
- MakeValidLatexString
 - Ipopt::AmplOptionsList, [74](#)
 - Ipopt::RegisteredOption, [685](#)
- MapStringSetting
 - Ipopt::RegisteredOption, [685](#)
- MapStringSettingToEnum
 - Ipopt::RegisteredOption, [685](#)
- match_strat_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
 - Ipopt::PardisoSolverInterface, [629](#)
- matrices_
 - Ipopt::SumMatrix, [784](#)
 - Ipopt::SumSymMatrix, [790](#)
- matrices_valid_
 - Ipopt::CompoundMatrix, [155](#)
 - Ipopt::CompoundSymMatrix, [165](#)
- MatricesValid
 - Ipopt::CompoundMatrix, [155](#)
 - Ipopt::CompoundSymMatrix, [164](#)
- Matrix
 - Ipopt::Matrix, [535](#)
- matrix_
 - Ipopt::ScaledMatrix, [728](#)
 - Ipopt::SymScaledMatrix, [803](#)
- matrix_dup
 - ma77_info_d, [504](#)
 - ma97_info, [524](#)
- matrix_file_number_
 - Ipopt::IterativeWsmpSolverInterface, [436](#)
 - Ipopt::WsmpSolverInterface, [911](#)
- matrix_format_
 - Ipopt::TSymLinearSolver, [883](#)
- matrix_missing_diag
 - ma97_info, [524](#)
- matrix_outrange
 - ma77_info_d, [505](#)
 - ma97_info, [524](#)
- matrix_rank
 - ma77_info_d, [504](#)
 - ma86_info_d, [514](#)
 - ma97_info, [524](#)
- MatrixFormat
 - Ipopt::IterativePardisoSolverInterface, [422](#)
 - Ipopt::IterativeWsmpSolverInterface, [434](#)
 - Ipopt::Ma27TSolverInterface, [488](#)
 - Ipopt::Ma57TSolverInterface, [498](#)
 - Ipopt::Ma77SolverInterface, [511](#)
 - Ipopt::Ma86SolverInterface, [520](#)
 - Ipopt::Ma97SolverInterface, [530](#)
 - Ipopt::MumpsSolverInterface, [564](#)
 - Ipopt::PardisoSolverInterface, [628](#)
 - Ipopt::SparseSymLinearSolverInterface, [752](#)
 - Ipopt::WsmpSolverInterface, [909](#)
- MatrixSpace
 - Ipopt::MatrixSpace, [541](#)
- Max
 - Ipopt, [51](#), [52](#)

- Ipopt::Vector, [893](#)
- max_alpha_x_
 - Ipopt::CGPenaltyData, [128](#)
- max_cache_size_
 - Ipopt::CachedResults, [117](#)
- max_cache_tag_
 - Ipopt::Vector, [898](#)
- max_cpu_time_
 - Ipopt::OptimalityErrorConvergenceCheck, [597](#)
- max_filter_resets_
 - Ipopt::FilterLSAceptor, [254](#)
- max_iterations_
 - Ipopt::OptimalityErrorConvergenceCheck, [596](#)
- max_piece_number_
 - Ipopt::PiecewisePenalty, [658](#)
- max_rank_
 - Ipopt::LowRankSSAugSystemSolver, [474](#)
- max_ref_val
 - Ipopt::AdaptiveMuUpdate, [60](#)
- max_refinement_steps_
 - Ipopt::PDFullSpaceSolver, [635](#)
- max_soc_
 - Ipopt::CGPenaltyLSAceptor, [137](#)
 - Ipopt::FilterLSAceptor, [254](#)
 - Ipopt::PenaltyLSAceptor, [653](#)
- max_soft_resto_iters_
 - Ipopt::BacktrackingLineSearch, [103](#)
- MaxImpl
 - Ipopt::CompoundVector, [174](#)
 - Ipopt::DenseVector, [207](#)
 - Ipopt::Vector, [896](#)
- maxdepth
 - ma77_info_d, [505](#)
 - ma86_info_d, [514](#)
 - ma97_info, [524](#)
- maxfront
 - ma77_info_d, [505](#)
 - ma97_info, [524](#)
- maxfrt_
 - Ipopt::Ma27TSolverInterface, [491](#)
- Maximum_CpuTime_Exceeded
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- Maximum_Iterations_Exceeded
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- maximum_iters_
 - Ipopt::RestoConvergenceCheck, [696](#)
- maximum_resto_iters_
 - Ipopt::RestoConvergenceCheck, [696](#)
- maxit
 - ma77_control_d, [502](#)
- maxstore
 - ma77_control_d, [502](#)
- Mc19TSymScalingMethod
 - Ipopt::Mc19TSymScalingMethod, [543](#)
- mc19ad_t
 - HSLLoader.h, [989](#)
- mc68_control, [543](#)
 - f_array_in, [544](#)
 - f_array_out, [544](#)
 - hsl_mc68i.h, [957](#)
 - lp, [544](#)
 - min_l_workspace, [544](#)
 - mp, [544](#)
 - nemin, [544](#)
 - print_level, [544](#)
 - row_full_thresh, [544](#)
 - row_search, [544](#)
 - wp, [544](#)
- mc68_default_control
 - hsl_mc68i.h, [957](#), [958](#)
- mc68_default_control_t
 - HSLLoader.h, [989](#)
- mc68_info, [545](#)
 - duplicate, [545](#)
 - flag, [545](#)
 - hsl_mc68i.h, [957](#)
 - iostat, [545](#)
 - l_workspace, [546](#)
 - n_compressions, [545](#)
 - n_dense_rows, [546](#)
 - n_zero_eigs, [546](#)
 - out_range, [545](#)
 - stat, [545](#)
 - zb01_info, [546](#)
- mc68_order
 - hsl_mc68i.h, [957](#), [958](#)
- mc68_order_t
 - HSLLoader.h, [989](#)
- mehrotra_algorithm_
 - Ipopt::IpoptAlgorithm, [338](#)
 - Ipopt::PDSearchDirCalculator, [645](#)
- mem_percent_
 - Ipopt::MumpsSolverInterface, [565](#)
- meminc_factor_
 - Ipopt::Ma27TSolverInterface, [490](#)
- Message
 - Ipopt::IpoptException, [391](#)
- Min
 - Ipopt, [51](#), [52](#)
 - Ipopt::Vector, [894](#)
- min_alpha_primal_
 - Ipopt::CGPenaltyLSAceptor, [136](#)
- min_cache_tag_
 - Ipopt::Vector, [898](#)
- min_l_workspace
 - mc68_control, [544](#)

- min_piece_penalty_
 - Ipopt::PiecewisePenalty, [657](#)
- min_ref_val
 - Ipopt::AdaptiveMuUpdate, [60](#)
- min_refinement_steps_
 - Ipopt::PDFullSpaceSolver, [634](#)
- MinC_1NrmRestorationPhase
 - Ipopt::MinC_1NrmRestorationPhase, [548](#)
- MinImpl
 - Ipopt::CompoundVector, [174](#)
 - Ipopt::DenseVector, [207](#)
 - Ipopt::Vector, [897](#)
- minstore
 - ma77_info_d, [505](#)
- modify_hessian_with_slacks_
 - Ipopt::InexactPDSolver, [316](#)
- MonotoneMuUpdate
 - Ipopt::MonotoneMuUpdate, [551](#)
- mp
 - mc68_control, [544](#)
- msg_
 - Ipopt::IpoptException, [391](#)
- mu_allow_fast_monotone_decrease_
 - Ipopt::MonotoneMuUpdate, [552](#)
- mu_init_
 - Ipopt::DefaultIterateInitializer, [187](#)
 - Ipopt::MonotoneMuUpdate, [552](#)
- mu_initialized_
 - Ipopt::IpoptData, [387](#)
- mu_linear_decrease_factor_
 - Ipopt::AdaptiveMuUpdate, [61](#)
 - Ipopt::MonotoneMuUpdate, [552](#)
- mu_max_
 - Ipopt::AdaptiveMuUpdate, [60](#)
- mu_max_fact_
 - Ipopt::AdaptiveMuUpdate, [60](#)
- mu_min_
 - Ipopt::AdaptiveMuUpdate, [60](#)
- mu_min_default_
 - Ipopt::AdaptiveMuUpdate, [60](#)
- mu_superlinear_decrease_power_
 - Ipopt::AdaptiveMuUpdate, [61](#)
 - Ipopt::MonotoneMuUpdate, [552](#)
- mu_target_
 - Ipopt::AdaptiveMuUpdate, [60](#)
 - Ipopt::IpoptCalculatedQuantities, [366](#)
 - Ipopt::MonotoneMuUpdate, [553](#)
 - Ipopt::OptimalityErrorConvergenceCheck, [597](#)
- mu_update_
 - Ipopt::IpoptAlgorithm, [337](#)
- MuInitialized
 - Ipopt::IpoptData, [382](#)
- MuOracle
 - Ipopt::MuOracle, [567](#)
- MuUpdate
 - Ipopt::MuUpdate, [569](#)
- mult_diverg_feasibility_tol_
 - Ipopt::CGPenaltyLSAcceptor, [137](#)
 - Ipopt::CGPerturbationHandler, [146](#)
- mult_diverg_y_tol_
 - Ipopt::CGPenaltyLSAcceptor, [137](#)
- mult_g
 - IpStdCInterface.h, [1006](#)
- mult_x_L
 - IpStdCInterface.h, [1006](#)
- mult_x_U
 - IpStdCInterface.h, [1006](#)
- MultVector
 - Ipopt::Matrix, [536](#)
- MultVectorImpl
 - Ipopt::CompoundMatrix, [154](#)
 - Ipopt::CompoundSymMatrix, [163](#)
 - Ipopt::DenseGenMatrix, [192](#)
 - Ipopt::DenseSymMatrix, [198](#)
 - Ipopt::DiagMatrix, [218](#)
 - Ipopt::ExpandedMultiVectorMatrix, [230](#)
 - Ipopt::ExpansionMatrix, [236](#)
 - Ipopt::GenTMatrix, [267](#)
 - Ipopt::IdentityMatrix, [279](#)
 - Ipopt::LowRankUpdateSymMatrix, [480](#)
 - Ipopt::Matrix, [537](#)
 - Ipopt::MultiVectorMatrix, [557](#)
 - Ipopt::ScaledMatrix, [727](#)
 - Ipopt::SumMatrix, [783](#)
 - Ipopt::SumSymMatrix, [789](#)
 - Ipopt::SymScaledMatrix, [802](#)
 - Ipopt::SymTMatrix, [810](#)
 - Ipopt::TransposeMatrix, [860](#)
 - Ipopt::ZeroMatrix, [914](#)
 - Ipopt::ZeroSymMatrix, [918](#)
- MultiSolve
 - Ipopt::AugSystemSolver, [95](#)
 - Ipopt::GenAugSystemSolver, [259](#)
 - Ipopt::GenKKTsSolverInterface, [263](#)
 - Ipopt::IterativePardisoSolverInterface, [422](#)
 - Ipopt::IterativeWsmpSolverInterface, [434](#)
 - Ipopt::Ma27TSolverInterface, [487](#)
 - Ipopt::Ma57TSolverInterface, [497](#)
 - Ipopt::Ma77SolverInterface, [510](#)
 - Ipopt::Ma86SolverInterface, [519](#)
 - Ipopt::Ma97SolverInterface, [529](#)
 - Ipopt::MumpsSolverInterface, [563](#)
 - Ipopt::PardisoSolverInterface, [627](#)
 - Ipopt::SparseSymLinearSolverInterface, [751](#)
 - Ipopt::StdAugSystemSolver, [762](#)
 - Ipopt::SymLinearSolver, [794](#)
 - Ipopt::TSymLinearSolver, [880](#)
 - Ipopt::WsmpSolverInterface, [908](#)

- MultiVectorMatrix
 - Ipopt::MultiVectorMatrix, [555](#)
- MultiVectorMatrixOwnerSpace
 - Ipopt::MultiVectorMatrix, [557](#)
- MultiVectorMatrixSpace
 - Ipopt::MultiVectorMatrixSpace, [559](#)
- multiplier
 - ma77_control_d, [502](#)
 - ma97_control_d, [522](#)
- MultipliersDiverged
 - Ipopt::CGPenaltyLSAcceptor, [135](#)
- mumps_dep_tol_
 - Ipopt::MumpsSolverInterface, [566](#)
- mumps_permuting_scaling_
 - Ipopt::MumpsSolverInterface, [565](#)
- mumps_pivot_order_
 - Ipopt::MumpsSolverInterface, [566](#)
- mumps_ptr_
 - Ipopt::MumpsSolverInterface, [565](#)
- mumps_scaling_
 - Ipopt::MumpsSolverInterface, [566](#)
- MumpsSolverInterface
 - Ipopt::MumpsSolverInterface, [563](#)
- N
- NEVER_MONOTONE_MODE
 - Ipopt::AdaptiveMuUpdate, [59](#)
- NM_NORM_1
 - Ipopt::QualityFunctionMuOracle, [668](#)
- NM_NORM_2
 - Ipopt::QualityFunctionMuOracle, [668](#)
- NM_NORM_2_SQUARED
 - Ipopt::QualityFunctionMuOracle, [668](#)
- NM_NORM_MAX
 - Ipopt::QualityFunctionMuOracle, [668](#)
- NO_CORRECTOR
 - Ipopt::FilterLSAcceptor, [250](#)
- NO_TEST
 - Ipopt::CGPerturbationHandler, [142](#)
 - Ipopt::PDPerturbationHandler, [638](#)
 - Ipopt::TNLPAdapter, [842](#)
- NON_LINEAR
 - Ipopt::TNLP, [832](#)
- NONE
 - Ipopt::DenseGenMatrix, [190](#)
- NONLINEAR_VARS
 - Ipopt, [46](#)
- NORM_1
 - Ipopt, [45](#)
- NORM_2
 - Ipopt, [45](#)
- NORM_MAX
 - Ipopt, [45](#)
- NOT_DEGENERATE
 - Ipopt::CGPerturbationHandler, [142](#)
 - Ipopt::PDPerturbationHandler, [638](#)
- NOT_YET_DETERMINED
 - Ipopt::CGPerturbationHandler, [142](#)
 - Ipopt::PDPerturbationHandler, [638](#)
- NT_All
 - Ipopt::Observer, [591](#)
- NT_BeingDestroyed
 - Ipopt::Observer, [591](#)
- NT_Changed
 - Ipopt::Observer, [591](#)
- n_comp_
 - Ipopt::QualityFunctionMuOracle, [672](#)
- n_compressions
 - mc68_info, [545](#)
- n_con_
 - Ipopt::StdInterfaceTNLP, [772](#)
- n_dense_rows
 - mc68_info, [546](#)
- n_dual_
 - Ipopt::QualityFunctionMuOracle, [672](#)
- n_filter_resets_
 - Ipopt::FilterLSAcceptor, [256](#)
- n_full_g_
 - Ipopt::TNLPAdapter, [847](#)
- n_full_x_
 - Ipopt::TNLPAdapter, [847](#)
- n_g_skip_
 - Ipopt::TNLPReducer, [857](#)
- n_pri_
 - Ipopt::QualityFunctionMuOracle, [672](#)
- n_var_
 - Ipopt::StdInterfaceTNLP, [772](#)
- n_x_fix_
 - Ipopt::TNLPReducer, [858](#)
- n_x_fixed_
 - Ipopt::TNLPAdapter, [848](#)
- n_xL_skip_
 - Ipopt::TNLPReducer, [858](#)
- n_xU_skip_
 - Ipopt::TNLPReducer, [858](#)
- n_zero_eigs
 - mc68_info, [546](#)
- NCols
 - Ipopt::Matrix, [536](#)
 - Ipopt::MatrixSpace, [541](#)
- nCols_
 - Ipopt::MatrixSpace, [542](#)
- NCompSpaces
 - Ipopt::CompoundVectorSpace, [179](#)
- NComps
 - Ipopt::CompoundVector, [172](#)
- NComps_Cols
 - Ipopt::CompoundMatrix, [154](#)

- Ipopt::CompoundMatrixSpace, 159
- NComps_Dim
 - Ipopt::CompoundSymMatrix, 163
 - Ipopt::CompoundSymMatrixSpace, 167
- NComps_Rows
 - Ipopt::CompoundMatrix, 154
 - Ipopt::CompoundMatrixSpace, 159
- NError
 - Ipopt::AmplOptionsList::PrivatInfo, 661
- NLP
 - Ipopt::NLP, 572
- NLP_scaling
 - Ipopt::IpoptNLP, 398
- NLPBoundsRemover
 - Ipopt::NLPBoundsRemover, 577
- NLPScalingObject
 - Ipopt::NLPScalingObject, 584
- NRows
 - Ipopt::Matrix, 536
 - Ipopt::MatrixSpace, 541
- nRows_
 - Ipopt::MatrixSpace, 541
- NTerms
 - Ipopt::SumMatrix, 783
 - Ipopt::SumMatrixSpace, 786
 - Ipopt::SumSymMatrix, 789
 - Ipopt::SumSymMatrixSpace, 792
- Name
 - Ipopt::Journal, 438
 - Ipopt::RegisteredOption, 681
- name_
 - Ipopt::Journal, 439
 - Ipopt::RegisteredOption, 685
- nb
 - ma86_control_d, 513
- nb54
 - ma77_control_d, 502
- nb64
 - ma77_control_d, 502
- nbi
 - ma77_control_d, 502
 - ma86_control_d, 513
- ncomp_spaces_
 - Ipopt::CompoundSymMatrixSpace, 168
 - Ipopt::CompoundVectorSpace, 179
- ncomps_cols_
 - Ipopt::CompoundMatrixSpace, 159
- ncomps_rows_
 - Ipopt::CompoundMatrixSpace, 159
- ndelay
 - ma77_info_d, 505
- ndim_
 - Ipopt::Ma77SolverInterface, 511
 - Ipopt::Ma86SolverInterface, 520
- Ipopt::Ma97SolverInterface, 530
- Neg_Omega_c_plus_D_c
 - Ipopt::AugRestoSystemSolver, 91
- Neg_Omega_d_plus_D_d
 - Ipopt::AugRestoSystemSolver, 91
- neg_curv_test_tol_
 - Ipopt::PDFullSpaceSolver, 635
- neg_omega_c_plus_D_c_cache_
 - Ipopt::AugRestoSystemSolver, 91
- neg_omega_d_plus_D_d_cache_
 - Ipopt::AugRestoSystemSolver, 91
- negEvalsCorrection_
 - Ipopt::LowRankSSAugSystemSolver, 476
- negevals_
 - Ipopt::IterativePardisoSolverInterface, 423
 - Ipopt::Ma27TSolverInterface, 489
 - Ipopt::Ma57TSolverInterface, 498
 - Ipopt::MumpsSolverInterface, 565
 - Ipopt::PardisoSolverInterface, 629
 - Ipopt::WsmvSolverInterface, 911
- nele_hess
 - IpStdCInterface.h, 1004
- nele_hess_
 - Ipopt::StdInterfaceTNLP, 773
- nele_jac
 - IpStdCInterface.h, 1004
- nele_jac_
 - Ipopt::StdInterfaceTNLP, 773
- nemin
 - ma77_control_d, 502
 - ma86_control_d, 513
 - ma97_control_d, 521
 - mc68_control, 544
- nerror_
 - Ipopt::AmplOptionsList::PrivatInfo, 662
 - Ipopt::AmplTNLP, 86
- nerror_ok
 - Ipopt::AmplTNLP, 85
- never_try_pure_Newton_
 - Ipopt::CGPenaltyData, 128
- never_use_fact_cgpen_direction_
 - Ipopt::CGSearchDirCalculator, 150
- never_use_piecewise_penalty_ls_
 - Ipopt::CGPenaltyLSAcceptor, 138
- NeverRestorationPhase
 - Ipopt::BacktrackingLSAcceptor, 111
 - Ipopt::CGPenaltyLSAcceptor, 134
- NeverTryPureNewton
 - Ipopt::CGPenaltyData, 126
- NewFixedMu
 - Ipopt::AdaptiveMuUpdate, 60
- newton_step_
 - Ipopt::InexactDoglegNormalStep, 296
- next_compute_normal

- [Ipopt::InexactData](#), 293
- [next_compute_normal_](#)
 - [Ipopt::InexactData](#), 294
- [next_counter_](#)
 - [Ipopt::RegisteredOptions](#), 692
- [nfactor](#)
 - [ma77_info_d](#), 505
- [nflops](#)
 - [ma77_info_d](#), 505
- [nio_read](#)
 - [ma77_info_d](#), 506
- [nio_write](#)
 - [ma77_info_d](#), 506
- [niter](#)
 - [ma77_info_d](#), 505
- [nkeywds_](#)
 - [Ipopt::AmplOptionsList](#), 74
- [nlp](#)
 - [Ipopt::NLPBoundsRemover](#), 580
 - [Ipopt::OrigIpoptNLP](#), 614
- [nlp_](#)
 - [Ipopt::EquilibrationScaling](#), 225
 - [Ipopt::GradientScaling](#), 274
 - [Ipopt::NLPBoundsRemover](#), 580
 - [Ipopt::OrigIpoptNLP](#), 616
 - [Ipopt::UserScaling](#), 887
- [nlp_adapter_](#)
 - [Ipopt::IpoptApplication](#), 345
- [nlp_lower_bound_inf_](#)
 - [Ipopt::TNLPAdapter](#), 845
- [nlp_scaling_](#)
 - [Ipopt::IpoptNLP](#), 398
- [nlp_upper_bound_inf_](#)
 - [Ipopt::TNLPAdapter](#), 845
- [nnz_jac_g_orig_](#)
 - [Ipopt::TNLPReducer](#), 857
- [nnz_jac_g_reduced_](#)
 - [Ipopt::TNLPReducer](#), 858
- [nnz_jac_g_skipped_](#)
 - [Ipopt::TNLPReducer](#), 858
- [no_bounds_](#)
 - [Ipopt::AdaptiveMuUpdate](#), 63
- [NoNLPScalingObject](#)
 - [Ipopt::NoNLPScalingObject](#), 589
- [NonIpopt_Exception_Thrown](#)
 - [IpReturnCodes_inc.h](#), 998
 - [Ipopt](#), 49
- [non_const_vecs_](#)
 - [Ipopt::MultiVectorMatrix](#), 558
- [non_const_x_](#)
 - [Ipopt::StdInterfaceTNLP](#), 774
- [nonZeros_](#)
 - [Ipopt::GenTMatrixSpace](#), 271
 - [Ipopt::SymTMatrixSpace](#), 813
- [nonconst_matrix_](#)
 - [Ipopt::ScaledMatrix](#), 729
 - [Ipopt::SymScaledMatrix](#), 803
- [nonmonotone_pen_update_counter_](#)
 - [Ipopt::CGSearchDirCalculator](#), 150
- [Nonzeros](#)
 - [Ipopt::GenTMatrix](#), 267
 - [Ipopt::GenTMatrixSpace](#), 271
 - [Ipopt::SymTMatrix](#), 809
 - [Ipopt::SymTMatrixSpace](#), 812
- [nonzeros_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 423
 - [Ipopt::Ma27TSolverInterface](#), 489
 - [Ipopt::Ma57TSolverInterface](#), 498
 - [Ipopt::PardisoSolverInterface](#), 628
 - [Ipopt::WsmvSolverInterface](#), 910
- [nonzeros_compressed_](#)
 - [Ipopt::TripletToCSRConverter](#), 874
 - [Ipopt::TSymLinearSolver](#), 882
- [nonzeros_triplet_](#)
 - [Ipopt::TripletToCSRConverter](#), 874
 - [Ipopt::TSymLinearSolver](#), 882
- [NormEnum](#)
 - [Ipopt::QualityFunctionMuOracle](#), 668
- [normal_pardiso_iter_coarse_size_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_pardiso_iter_dropping_factor_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_pardiso_iter_dropping_factor_used_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_pardiso_iter_dropping_schur_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_pardiso_iter_dropping_schur_used_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 426
- [normal_pardiso_iter_inverse_norm_factor_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_pardiso_iter_max_levels_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_pardiso_iter_max_row_fill_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_pardiso_iter_relative_tol_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_pardiso_max_iter_](#)
 - [Ipopt::IterativePardisoSolverInterface](#), 425
- [normal_s](#)
 - [Ipopt::InexactData](#), 292
- [normal_s_](#)
 - [Ipopt::InexactData](#), 293
- [normal_step_calculator_](#)
 - [Ipopt::InexactSearchDirCalculator](#), 326
- [normal_tester_](#)
 - [Ipopt::InexactDoglegNormalStep](#), 296
 - [Ipopt::IterativePardisoSolverInterface](#), 427
- [normal_x](#)

- Ipopt::InexactData, [292](#)
- normal_x_
 - Ipopt::InexactData, [293](#)
- Not_Enough_Degrees_Of_Freedom
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- Notify
 - Ipopt::Subject, [780](#)
- NotifyType
 - Ipopt::Observer, [591](#)
- Nrm2
 - Ipopt::Vector, [892](#)
- nrm2_cache_tag_
 - Ipopt::Vector, [898](#)
- Nrm2Impl
 - Ipopt::CompoundVector, [173](#)
 - Ipopt::DenseVector, [206](#)
 - Ipopt::Vector, [895](#)
- nsteps_
 - Ipopt::Ma27TSolverInterface, [491](#)
- nsup
 - ma77_info_d, [505](#)
- nterms_
 - Ipopt::SumMatrixSpace, [787](#)
 - Ipopt::SumSymMatrixSpace, [792](#)
- ntwo
 - ma77_info_d, [505](#)
- nu_
 - Ipopt::InexactLSAcceptor, [304](#)
 - Ipopt::PenaltyLSAcceptor, [654](#)
- nu_inc_
 - Ipopt::InexactLSAcceptor, [303](#)
 - Ipopt::PenaltyLSAcceptor, [653](#)
- nu_init_
 - Ipopt::InexactLSAcceptor, [302](#)
 - Ipopt::PenaltyLSAcceptor, [653](#)
- nu_low_
 - Ipopt::InexactLSAcceptor, [304](#)
- nu_low_fact_
 - Ipopt::InexactLSAcceptor, [303](#)
- nu_low_init_
 - Ipopt::InexactLSAcceptor, [303](#)
- nu_update_inf_skip_tol_
 - Ipopt::InexactLSAcceptor, [303](#)
- num_adjusted_slack_s_L_
 - Ipopt::IpoptCalculatedQuantities, [367](#)
- num_adjusted_slack_s_U_
 - Ipopt::IpoptCalculatedQuantities, [367](#)
- num_adjusted_slack_x_L_
 - Ipopt::IpoptCalculatedQuantities, [367](#)
- num_adjusted_slack_x_U_
 - Ipopt::IpoptCalculatedQuantities, [367](#)
- num_constr_evals_
 - Ipopt::SolveStatistics, [745](#)
- num_constr_jac_evals_
 - Ipopt::SolveStatistics, [746](#)
- num_delay
 - ma86_info_d, [514](#)
 - ma97_info, [524](#)
- num_doubles_
 - Ipopt::TripletToCSRConverter, [874](#)
- num_factor
 - ma86_info_d, [515](#)
 - ma97_info, [524](#)
- num_file
 - ma77_info_d, [506](#)
- num_flops
 - ma86_info_d, [515](#)
 - ma97_info, [524](#)
- num_hess_evals_
 - Ipopt::SolveStatistics, [746](#)
- num_iters_
 - Ipopt::SolveStatistics, [745](#)
- num_linear_variables_
 - Ipopt::TNLPAdapter, [846](#)
- num_neg
 - ma77_info_d, [505](#)
 - ma86_info_d, [515](#)
 - ma97_info, [524](#)
- num_neg_evals_
 - Ipopt::LowRankAugSystemSolver, [469](#)
 - Ipopt::LowRankSSAugSystemSolver, [476](#)
- num_nodes
 - ma86_info_d, [515](#)
- num_nothresh
 - ma77_info_d, [505](#)
 - ma86_info_d, [515](#)
- num_obj_evals_
 - Ipopt::SolveStatistics, [745](#)
- num_obj_grad_evals_
 - Ipopt::SolveStatistics, [745](#)
- num_perturbed
 - ma77_info_d, [505](#)
 - ma86_info_d, [515](#)
- num_refs_max_
 - Ipopt::AdaptiveMuUpdate, [62](#)
- num_sup
 - ma97_info, [524](#)
- num_two
 - ma86_info_d, [515](#)
 - ma97_info, [524](#)
- Number
 - Ipopt, [44](#)
 - IpStdCInterface.h, [1001](#)
- Number_Option
 - Ipopt::AmplOptionsList, [73](#)
- Number_Type
 - Ipopt::AmplSuffixHandler, [76](#)

- NumberOfAmplOptions
 - Ipopt::AmplOptionsList, [74](#)
- NumberOfEvaluations
 - Ipopt::SolveStatistics, [744](#)
- NumberOfNegEvals
 - Ipopt::AugRestoSystemSolver, [90](#)
 - Ipopt::AugSystemSolver, [95](#)
 - Ipopt::GenAugSystemSolver, [259](#)
 - Ipopt::GenKKTsSolverInterface, [264](#)
 - Ipopt::IterativePardisoSolverInterface, [422](#)
 - Ipopt::IterativeWsmplSolverInterface, [434](#)
 - Ipopt::LowRankAugSystemSolver, [466](#)
 - Ipopt::LowRankSSAugSystemSolver, [473](#)
 - Ipopt::Ma27TSolverInterface, [487](#)
 - Ipopt::Ma57TSolverInterface, [497](#)
 - Ipopt::Ma77SolverInterface, [510](#)
 - Ipopt::Ma86SolverInterface, [519](#)
 - Ipopt::Ma97SolverInterface, [529](#)
 - Ipopt::MumpsSolverInterface, [563](#)
 - Ipopt::PardisoSolverInterface, [627](#)
 - Ipopt::SparseSymLinearSolverInterface, [752](#)
 - Ipopt::StdAugSystemSolver, [763](#)
 - Ipopt::SymLinearSolver, [794](#)
 - Ipopt::TSymLinearSolver, [880](#)
 - Ipopt::WsmplSolverInterface, [908](#)
- numdelay_
 - Ipopt::Ma97SolverInterface, [531](#)
- numeric_meta_data_
 - Ipopt::DenseVectorSpace, [213](#)
- NumericMetaDataMapType
 - Ipopt, [45](#)
 - Ipopt::TNLP, [831](#)
- numneg_
 - Ipopt::Ma77SolverInterface, [511](#)
 - Ipopt::Ma86SolverInterface, [520](#)
 - Ipopt::Ma97SolverInterface, [531](#)
- nwd_read
 - ma77_info_d, [506](#)
- nwd_write
 - ma77_info_d, [506](#)
- nz_full_h_
 - Ipopt::TNLPAdapter, [848](#)
- nz_full_jac_g_
 - Ipopt::TNLPAdapter, [847](#)
- nz_h_
 - Ipopt::TNLPAdapter, [848](#)
- nz_h_full_
 - Ipopt::AmplTNLP, [85](#)
- nz_jac_c_
 - Ipopt::TNLPAdapter, [847](#)
- nz_jac_c_no_extra_
 - Ipopt::TNLPAdapter, [847](#)
- nz_jac_d_
 - Ipopt::TNLPAdapter, [847](#)
- O
 - ONLY_SECOND_ORDER_TEST
 - Ipopt::TNLPAdapter, [842](#)
 - ORDER_AMD
 - Ipopt::Ma77SolverInterface, [509](#)
 - Ipopt::Ma86SolverInterface, [518](#)
 - Ipopt::Ma97SolverInterface, [528](#)
 - ORDER_AUTO
 - Ipopt::Ma86SolverInterface, [518](#)
 - Ipopt::Ma97SolverInterface, [528](#)
 - ORDER_BEST
 - Ipopt::Ma97SolverInterface, [528](#)
 - ORDER_MATCHED_AMD
 - Ipopt::Ma97SolverInterface, [528](#)
 - ORDER_MATCHED_AUTO
 - Ipopt::Ma97SolverInterface, [528](#)
 - ORDER_MATCHED_METIS
 - Ipopt::Ma97SolverInterface, [528](#)
 - ORDER_METIS
 - Ipopt::Ma77SolverInterface, [509](#)
 - Ipopt::Ma86SolverInterface, [518](#)
 - Ipopt::Ma97SolverInterface, [528](#)
 - ORIGINAL
 - Ipopt::IterationOutput, [416](#)
 - OT_Integer
 - Ipopt, [48](#)
 - OT_Number
 - Ipopt, [48](#)
 - OT_String
 - Ipopt, [48](#)
 - OT_Unknown
 - Ipopt, [48](#)
 - OTHER_SATISFIED
 - Ipopt::IterativeSolverTerminationTester, [429](#)
 - OUT_OF_MEMORY
 - Ipopt, [48](#)
 - obj_max_inc_
 - Ipopt::FilterLSAcceptor, [254](#)
 - obj_scaling
 - IpStdCInterface.h, [1005](#)
 - obj_scaling_
 - Ipopt::StdInterfaceTNLP, [774](#)
 - obj_scaling_factor_
 - Ipopt::StandardScalingBase, [759](#)
 - obj_sign_
 - Ipopt::AmplTNLP, [85](#)
 - obj_sol_
 - Ipopt::AmplTNLP, [86](#)
 - Ipopt::StdInterfaceTNLP, [775](#)
 - obj_val
 - IpStdCInterface.h, [1006](#)
 - obj_val_
 - Ipopt::SolveStatistics, [746](#)
 - ObjectChanged

- Ipopt::TaggedObject, 816
- Objective_Source
 - Ipopt::AmplSuffixHandler, 76
- objective_depends_on_mu
 - Ipopt::IpoptNLP, 397
 - Ipopt::RestoIpoptNLP, 704
- objval_called_with_current_x_
 - Ipopt::AmplTNLP, 86
- Observer
 - Ipopt::Observer, 591
- observers_
 - Ipopt::Subject, 781
- offset_
 - Ipopt::TripletToCSRConverter, 873
- Oinfo_ptr_
 - Ipopt::AmplTNLP, 86
- old_w_
 - Ipopt::StdAugSystemSolver, 766
- omega_max_
 - Ipopt::InexactDoglegNormalStep, 296
- Open
 - Ipopt::FileJournal, 241
- OpenOutputFile
 - Ipopt::IpoptApplication, 342
- operator<
 - Ipopt, 51
 - Ipopt::SmartPtr, 741
 - Ipopt::TripletToCSRConverter::TripletEntry, 866
- operator<=
 - Ipopt, 51
- operator>
 - Ipopt, 51
- operator>=
 - Ipopt, 51
- operator*
 - Ipopt::SmartPtr, 739
- operator+
 - Ipopt, 51
- operator->
 - Ipopt::SmartPtr, 739
- operator=
 - Ipopt::AdaptiveMuUpdate, 59
 - Ipopt::AlgorithmBuilder, 65
 - Ipopt::AlgorithmStrategyObject, 69
 - Ipopt::AmplOptionsList, 74
 - Ipopt::AmplOptionsList::AmplOption, 71
 - Ipopt::AmplSuffixHandler, 77
 - Ipopt::AmplTNLP, 84
 - Ipopt::AugRestoSystemSolver, 91
 - Ipopt::AugSystemSolver, 95
 - Ipopt::BacktrackingLineSearch, 101
 - Ipopt::BacktrackingLSAccepter, 111
 - Ipopt::CachedResults, 116
 - Ipopt::CGPenaltyCq, 120
 - Ipopt::CGPenaltyData, 127
 - Ipopt::CGPenaltyLSAccepter, 134
 - Ipopt::CGPerturbationHandler, 143
 - Ipopt::CGSearchDirCalculator, 149
 - Ipopt::CompoundMatrix, 155
 - Ipopt::CompoundMatrixSpace, 159
 - Ipopt::CompoundSymMatrix, 164
 - Ipopt::CompoundSymMatrixSpace, 168
 - Ipopt::CompoundVector, 175
 - Ipopt::CompoundVectorSpace, 179
 - Ipopt::ConvergenceCheck, 182
 - Ipopt::DefaultIterateInitializer, 186
 - Ipopt::DenseGenMatrix, 193
 - Ipopt::DenseSymMatrix, 199
 - Ipopt::DenseVector, 208
 - Ipopt::DependentResult, 215
 - Ipopt::DiagMatrix, 219
 - Ipopt::DiagMatrixSpace, 221
 - Ipopt::EqMultiplierCalculator, 223
 - Ipopt::EquilibrationScaling, 225
 - Ipopt::ExactHessianUpdater, 227
 - Ipopt::ExpandedMultiVectorMatrix, 231
 - Ipopt::ExpansionMatrix, 237
 - Ipopt::FileJournal, 242
 - Ipopt::Filter, 244
 - Ipopt::FilterEntry, 246
 - Ipopt::FilterLSAccepter, 252
 - Ipopt::GenAugSystemSolver, 259
 - Ipopt::GenTMatrix, 268
 - Ipopt::GradientScaling, 274
 - Ipopt::HessianUpdater, 276
 - Ipopt::IdentityMatrix, 279
 - Ipopt::IdentityMatrixSpace, 281
 - Ipopt::InexactAlgorithmBuilder, 283
 - Ipopt::InexactCq, 287
 - Ipopt::InexactData, 293
 - Ipopt::InexactDoglegNormalStep, 296
 - Ipopt::InexactLSAccepter, 302
 - Ipopt::InexactNewtonNormalStep, 307
 - Ipopt::InexactNormalStepCalculator, 309
 - Ipopt::InexactNormalTerminationTester, 312
 - Ipopt::InexactPDSolver, 315
 - Ipopt::InexactPDTerminationTester, 320
 - Ipopt::InexactSearchDirCalculator, 326
 - Ipopt::InexactTSymScalingMethod, 328
 - Ipopt::IpoptAdditionalCq, 330
 - Ipopt::IpoptAdditionalData, 332
 - Ipopt::IpoptAlgorithm, 336
 - Ipopt::IpoptApplication, 343
 - Ipopt::IpoptCalculatedQuantities, 364
 - Ipopt::IpoptData, 385
 - Ipopt::IpoptException, 391
 - Ipopt::IpoptNLP, 398
 - Ipopt::IterateInitializer, 400

- [Ipopt::IteratesVector](#), 411
- [Ipopt::IteratesVectorSpace](#), 414
- [Ipopt::IterationOutput](#), 417
- [Ipopt::IterativePardisoSolverInterface](#), 423
- [Ipopt::IterativeSolverTerminationTester](#), 430
- [Ipopt::IterativeWsmvSolverInterface](#), 434
- [Ipopt::Journal](#), 439
- [Ipopt::Journalist](#), 443
- [Ipopt::LeastSquareMultipliers](#), 445
- [Ipopt::LimMemQuasiNewtonUpdater](#), 451
- [Ipopt::LineSearch](#), 461
- [Ipopt::LoqoMuOracle](#), 463
- [Ipopt::LowRankAugSystemSolver](#), 467
- [Ipopt::LowRankSSAugSystemSolver](#), 473
- [Ipopt::LowRankUpdateSymMatrix](#), 480
- [Ipopt::LowRankUpdateSymMatrixSpace](#), 483
- [Ipopt::Ma27TSolverInterface](#), 488
- [Ipopt::Ma28TDependencyDetector](#), 493
- [Ipopt::Ma57TSolverInterface](#), 498
- [Ipopt::Matrix](#), 538
- [Ipopt::MatrixSpace](#), 541
- [Ipopt::Mc19TSymScalingMethod](#), 543
- [Ipopt::MinC_1NrmRestorationPhase](#), 548
- [Ipopt::MonotoneMuUpdate](#), 552
- [Ipopt::MultiVectorMatrix](#), 558
- [Ipopt::MumpsSolverInterface](#), 564
- [Ipopt::MuOracle](#), 568
- [Ipopt::MuUpdate](#), 570
- [Ipopt::NLP](#), 575
- [Ipopt::NLPBoundsRemover](#), 580
- [Ipopt::NLPScalingObject](#), 588
- [Ipopt::NoNLPScalingObject](#), 589
- [Ipopt::Observer](#), 592
- [Ipopt::OptimalityErrorConvergenceCheck](#), 595
- [Ipopt::OptionsList](#), 600
- [Ipopt::OptionsList::OptionValue](#), 604
- [Ipopt::OrigIpoptNLP](#), 615
- [Ipopt::OrigIterationOutput](#), 623
- [Ipopt::PardisoSolverInterface](#), 628
- [Ipopt::PDFullSpaceSolver](#), 633
- [Ipopt::PDPerturbationHandler](#), 640
- [Ipopt::PDSearchDirCalculator](#), 645
- [Ipopt::PDSystemSolver](#), 647
- [Ipopt::PenaltyLSAcceptor](#), 652
- [Ipopt::PiecewisePenalty](#), 657
- [Ipopt::PointPerturber](#), 660
- [Ipopt::ProbingMuOracle](#), 664
- [Ipopt::QualityFunctionMuOracle](#), 669
- [Ipopt::RestoConvergenceCheck](#), 695
- [Ipopt::RestoFilterConvergenceCheck](#), 698
- [Ipopt::RestoIpoptNLP](#), 709
- [Ipopt::RestoIterateInitializer](#), 715
- [Ipopt::RestoIterationOutput](#), 717
- [Ipopt::RestoPenaltyConvergenceCheck](#), 720
- [Ipopt::RestorationPhase](#), 722
- [Ipopt::RestoRestorationPhase](#), 724
- [Ipopt::ScaledMatrix](#), 728
- [Ipopt::ScaledMatrixSpace](#), 731
- [Ipopt::SearchDirectionCalculator](#), 733
- [Ipopt::SlackBasedTSymScalingMethod](#), 735
- [Ipopt::SmartPtr](#), 740
- [Ipopt::SolveStatistics](#), 745
- [Ipopt::StandardScalingBase](#), 759
- [Ipopt::StdAugSystemSolver](#), 763
- [Ipopt::StdInterfaceTNLP](#), 772
- [Ipopt::StreamJournal](#), 777
- [Ipopt::Subject](#), 781
- [Ipopt::SumMatrix](#), 784
- [Ipopt::SumMatrixSpace](#), 786
- [Ipopt::SumSymMatrix](#), 790
- [Ipopt::SymMatrixSpace](#), 799
- [Ipopt::SymScaledMatrix](#), 803
- [Ipopt::SymScaledMatrixSpace](#), 806
- [Ipopt::SymTMatrix](#), 810
- [Ipopt::TaggedObject](#), 816
- [Ipopt::TDependencyDetector](#), 818
- [Ipopt::TimedTask](#), 821
- [Ipopt::TimingStatistics](#), 826
- [Ipopt::TNLP](#), 835
- [Ipopt::TNLPAdapter](#), 845
- [Ipopt::TNLPReducer](#), 857
- [Ipopt::TransposeMatrix](#), 861
- [Ipopt::TransposeMatrixSpace](#), 864
- [Ipopt::TripletToCSRConverter](#), 873
- [Ipopt::TripletToCSRConverter::TripletEntry](#), 865
- [Ipopt::TSymDependencyDetector](#), 876
- [Ipopt::TSymLinearSolver](#), 881
- [Ipopt::TSymScalingMethod](#), 885
- [Ipopt::UserScaling](#), 887
- [Ipopt::Vector](#), 897
- [Ipopt::VectorSpace](#), 901
- [Ipopt::WarmStartIterateInitializer](#), 903
- [Ipopt::WsmvSolverInterface](#), 909
- [Ipopt::ZeroMatrix](#), 914
- [Ipopt::ZeroMatrixSpace](#), 916
- [Ipopt::ZeroSymMatrix](#), 919
- [Ipopt::ZeroSymMatrixSpace](#), 921
- operator==
 - [Ipopt](#), 50
 - [Ipopt::SmartPtr](#), 740
- OptimalityErrorConvergenceCheck
 - [Ipopt::OptimalityErrorConvergenceCheck](#), 595
- Optimize
 - [Ipopt::IpoptAlgorithm](#), 335
- OptimizeNLP
 - [Ipopt::IpoptApplication](#), 342
- OptimizeTNLP
 - [Ipopt::IpoptApplication](#), 342

- OptionValue
 - Ipopt::OptionsList::OptionValue, 603
- Options
 - Ipopt::AmplOptionsList::PrivatInfo, 661
 - Ipopt::IpoptApplication, 343
- options_
 - Ipopt::AmplOptionsList::PrivatInfo, 661
 - Ipopt::IpoptApplication, 344
 - Ipopt::OptionsList, 602
- OptionsList
 - Ipopt::OptionsList, 600
- order_
 - Ipopt::Ma86SolverInterface, 520
- order_opts
 - Ipopt::Ma77SolverInterface, 509
 - Ipopt::Ma86SolverInterface, 518
 - Ipopt::Ma97SolverInterface, 528
- ordering
 - ma97_control_d, 522
 - ma97_info, 524
- ordering_
 - Ipopt::Ma77SolverInterface, 512
 - Ipopt::Ma86SolverInterface, 521
 - Ipopt::Ma97SolverInterface, 531
- orig_aug_solver_
 - Ipopt::AugRestoSystemSolver, 92
- orig_constr_viol_tol_
 - Ipopt::RestoConvergenceCheck, 696
- orig_filter_ls_acceptor_
 - Ipopt::RestoFilterConvergenceCheck, 699
- orig_ip_cq_
 - Ipopt::RestoIpoptNLP, 709
- orig_ip_data_
 - Ipopt::RestoIpoptNLP, 709
- orig_ip_nlp_
 - Ipopt::RestoIpoptNLP, 709
- orig_matrix_
 - Ipopt::TransposeMatrix, 862
- orig_matrix_space_
 - Ipopt::TransposeMatrixSpace, 864
- orig_penalty_ls_acceptor_
 - Ipopt::RestoPenaltyConvergenceCheck, 720
- orig_x_L_
 - Ipopt::OrigIpoptNLP, 618
- orig_x_U_
 - Ipopt::OrigIpoptNLP, 619
- OrigIpCq
 - Ipopt::RestoIpoptNLP, 708
- OrigIpData
 - Ipopt::RestoIpoptNLP, 707
- OrigIpNLP
 - Ipopt::RestoIpoptNLP, 707
- OrigIpoptNLP
 - Ipopt::OrigIpoptNLP, 610
- OrigIterationOutput
 - Ipopt::OrigIterationOutput, 622
- OrigMatrix
 - Ipopt::TransposeMatrix, 860
- os_
 - Ipopt::StreamJournal, 777
- out_range
 - mc68_info, 545
- OutputDescription
 - Ipopt::RegisteredOption, 685
- OutputIteration
 - Ipopt::IpoptAlgorithm, 336
 - Ipopt::TimingStatistics, 825
- OutputIteration_
 - Ipopt::TimingStatistics, 827
- OutputLatexDescription
 - Ipopt::RegisteredOption, 685
- OutputLatexOptionDocumentation
 - Ipopt::RegisteredOptions, 692
- OutputOptionDocumentation
 - Ipopt::RegisteredOptions, 692
- OutputShortDescription
 - Ipopt::RegisteredOption, 685
- OverallAlgorithm
 - Ipopt::TimingStatistics, 824
- OverallAlgorithm_
 - Ipopt::TimingStatistics, 827
- owner_space_
 - Ipopt::CompoundMatrix, 155
 - Ipopt::CompoundSymMatrix, 164
 - Ipopt::CompoundVector, 176
 - Ipopt::DenseGenMatrix, 193
 - Ipopt::DenseSymMatrix, 199
 - Ipopt::DenseVector, 208
 - Ipopt::ExpandedMultiVectorMatrix, 231
 - Ipopt::ExpansionMatrix, 237
 - Ipopt::GenTMatrix, 268
 - Ipopt::IteratesVector, 411
 - Ipopt::LowRankUpdateSymMatrix, 480
 - Ipopt::Matrix, 539
 - Ipopt::MultiVectorMatrix, 558
 - Ipopt::ScaledMatrix, 729
 - Ipopt::SumMatrix, 784
 - Ipopt::SumSymMatrix, 790
 - Ipopt::SymMatrix, 797
 - Ipopt::SymScaledMatrix, 803
 - Ipopt::SymTMatrix, 810
 - Ipopt::Vector, 898
- OwnerSpace
 - Ipopt::Matrix, 537
 - Ipopt::Vector, 895
- OwnerSymMatrixSpace
 - Ipopt::SymMatrix, 797

- P
- PRIMAL_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, [100](#)
- PRIMAL_AND_FULL_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, [100](#)
- PRIMAL_DUAL_CORRECTOR
 - Ipopt::FilterLSAcceptor, [250](#)
- P_LowRank
 - Ipopt::LowRankUpdateSymMatrix, [479](#)
 - Ipopt::LowRankUpdateSymMatrixSpace, [483](#)
- P_LowRank_
 - Ipopt::LowRankUpdateSymMatrixSpace, [483](#)
- P_c_g_
 - Ipopt::TNLPAdapter, [850](#)
- P_c_g_space_
 - Ipopt::TNLPAdapter, [850](#)
- P_d_g_
 - Ipopt::TNLPAdapter, [851](#)
- P_d_g_space_
 - Ipopt::TNLPAdapter, [850](#)
- P_x_full_x_
 - Ipopt::TNLPAdapter, [850](#)
- P_x_full_x_space_
 - Ipopt::TNLPAdapter, [850](#)
- P_x_x_L_
 - Ipopt::TNLPAdapter, [850](#)
- P_x_x_L_space_
 - Ipopt::TNLPAdapter, [850](#)
- P_x_x_U_
 - Ipopt::TNLPAdapter, [850](#)
- P_x_x_U_space_
 - Ipopt::TNLPAdapter, [850](#)
- PDFullSpaceSolver
 - Ipopt::PDFullSpaceSolver, [633](#)
- PDPerturbationHandler
 - Ipopt::PDPerturbationHandler, [639](#)
- PDSearchDirCalculator
 - Ipopt::PDSearchDirCalculator, [644](#)
- PDSolver
 - Ipopt::PDSearchDirCalculator, [645](#)
- PDSysSystemSolver
 - Ipopt::PDSysSystemSolver, [647](#)
- PDSysSystemSolverSolveOnce
 - Ipopt::TimingStatistics, [825](#)
- PDSysSystemSolverSolveOnce_
 - Ipopt::TimingStatistics, [827](#)
- PDSysSystemSolverTotal
 - Ipopt::TimingStatistics, [825](#)
- PDSysSystemSolverTotal_
 - Ipopt::TimingStatistics, [827](#)
- PERM_
 - Ipopt::WsmpSolverInterface, [911](#)
- PT_
 - Ipopt::IterativePardisoSolverInterface, [426](#)
- Ipopt::PardisoSolverInterface, [629](#)
- ParExpansionMatrix
 - Ipopt::ExpansionMatrix, [237](#)
- ParGenMatrix
 - Ipopt::GenTMatrix, [268](#)
- ParVector
 - Ipopt::DenseVector, [208](#)
- pardiso_iter_coarse_size_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
- pardiso_iter_dropping_factor_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
- pardiso_iter_dropping_factor_used_
 - Ipopt::IterativePardisoSolverInterface, [425](#)
- pardiso_iter_dropping_schur_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
- pardiso_iter_dropping_schur_used_
 - Ipopt::IterativePardisoSolverInterface, [425](#)
- pardiso_iter_inverse_norm_factor_
 - Ipopt::IterativePardisoSolverInterface, [425](#)
- pardiso_iter_max_levels_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
- pardiso_iter_max_row_fill_
 - Ipopt::IterativePardisoSolverInterface, [425](#)
- pardiso_iter_relative_tol_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
- pardiso_iterative_
 - Ipopt::PardisoSolverInterface, [629](#)
- pardiso_max_droptol_corrections_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
 - Ipopt::PardisoSolverInterface, [629](#)
- pardiso_max_iter_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
- pardiso_redo_symbolic_fact_only_if_inertia_wrong_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
 - Ipopt::PardisoSolverInterface, [629](#)
- pardiso_repeated_perturbation_means_singular_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
 - Ipopt::PardisoSolverInterface, [629](#)
- PardisoLoader.h
 - LSL_PardisoLibraryName, [994](#)
 - LSL_isPardisoLoaded, [994](#)
 - LSL_loadPardisoLib, [993](#)
 - LSL_unloadPardisoLib, [994](#)
- PardisoMatchingStrategy
 - Ipopt::IterativePardisoSolverInterface, [421](#)
 - Ipopt::PardisoSolverInterface, [626](#)
- PardisoSolverInterface
 - Ipopt::PardisoSolverInterface, [627](#)
- Pd_L_
 - Ipopt::IpoptNLP, [396](#)
 - Ipopt::OrigIpoptNLP, [613](#)
 - Ipopt::RestIpoptNLP, [706](#)
- Pd_L_
 - Ipopt::OrigIpoptNLP, [618](#)

- Ipopt::RestIpoptNLP, 711
- Pd_U
 - Ipopt::IpoptNLP, 396
 - Ipopt::OrigIpoptNLP, 613
 - Ipopt::RestIpoptNLP, 707
- Pd_U_
 - Ipopt::OrigIpoptNLP, 618
 - Ipopt::RestIpoptNLP, 711
- pd_l_space_
 - Ipopt::OrigIpoptNLP, 616
 - Ipopt::RestIpoptNLP, 710
 - Ipopt::TNLPAdapter, 849
- pd_pert_c_
 - Ipopt::IpoptData, 389
- pd_pert_d_
 - Ipopt::IpoptData, 389
- pd_pert_s_
 - Ipopt::IpoptData, 389
- pd_pert_x_
 - Ipopt::IpoptData, 388
- pd_solver_
 - Ipopt::CGPenaltyLSAceptor, 138
 - Ipopt::CGSearchDirCalculator, 150
 - Ipopt::FilterLSAceptor, 256
 - Ipopt::PDSearchDirCalculator, 645
 - Ipopt::PenaltyLSAceptor, 655
 - Ipopt::ProbingMuOracle, 664
 - Ipopt::QualityFunctionMuOracle, 670
- pd_tester_
 - Ipopt::IterativePardisoSolverInterface, 427
- pd_u_space_
 - Ipopt::OrigIpoptNLP, 616
 - Ipopt::RestIpoptNLP, 710
 - Ipopt::TNLPAdapter, 849
- pen_curr_mu_
 - Ipopt::CGPenaltyLSAceptor, 136
- pen_des_fact_
 - Ipopt::CGSearchDirCalculator, 150
- pen_init_fac_
 - Ipopt::CGSearchDirCalculator, 150
- pen_r_
 - Ipopt::PiecewisePenEntry, 658
- pen_theta_max_
 - Ipopt::CGPenaltyLSAceptor, 136
- pen_theta_max_fact_
 - Ipopt::CGPenaltyLSAceptor, 136
- penalty_backward_
 - Ipopt::CGSearchDirCalculator, 150
- penalty_init_max_
 - Ipopt::CGSearchDirCalculator, 149
- penalty_init_min_
 - Ipopt::CGSearchDirCalculator, 149
- penalty_initialized_
 - Ipopt::CGPenaltyData, 128
- penalty_max_
 - Ipopt::CGPenaltyLSAceptor, 136
 - Ipopt::CGPerturbationHandler, 146
 - Ipopt::CGSearchDirCalculator, 149
- penalty_update_compl_tol_
 - Ipopt::CGPenaltyLSAceptor, 135
- penalty_update_infeasibility_tol_
 - Ipopt::CGPenaltyLSAceptor, 135
- PenaltyInitialized
 - Ipopt::CGPenaltyData, 127
- PenaltyLSAceptor
 - Ipopt::PenaltyLSAceptor, 651
- PerformDualStep
 - Ipopt::BacktrackingLineSearch, 102
- PerformGoldenSection
 - Ipopt::QualityFunctionMuOracle, 669
- PerformMagicStep
 - Ipopt::BacktrackingLineSearch, 102
- PerformRestoration
 - Ipopt::MinC_1NrmRestorationPhase, 548
 - Ipopt::RestorationPhase, 722
 - Ipopt::RestoRestorationPhase, 724
- pert_dir_
 - Ipopt::PointPerturber, 660
- perturb_always_cd_
 - Ipopt::CGPerturbationHandler, 146
 - Ipopt::PDPerturbationHandler, 642
- PerturbForSingularity
 - Ipopt::CGPerturbationHandler, 143
 - Ipopt::PDPerturbationHandler, 639
- PerturbForWrongInertia
 - Ipopt::CGPerturbationHandler, 143
 - Ipopt::PDPerturbationHandler, 639
- perturbHandler_
 - Ipopt::InexactPDSolver, 316
 - Ipopt::PDFullSpaceSolver, 634
- PiecewisePenEntry
 - Ipopt, 45
- PiecewisePenalty
 - Ipopt::PiecewisePenalty, 656
- PiecewisePenalty_
 - Ipopt::CGPenaltyLSAceptor, 138
- PiecewisePenalty_list_
 - Ipopt::PiecewisePenalty, 658
- piecewisepenalty_gamma_infeasi_
 - Ipopt::CGPenaltyLSAceptor, 136
- piecewisepenalty_gamma_obj_
 - Ipopt::CGPenaltyLSAceptor, 136
- pivot_
 - Ipopt::DenseGenMatrix, 194
- pivtol_
 - Ipopt::Ma27TSolverInterface, 489
 - Ipopt::Ma57TSolverInterface, 499
 - Ipopt::MumpsSolverInterface, 565

- pivotl_changed_
 - Ipopt::IterativeWsmSolverInterface, [436](#)
 - Ipopt::Ma27TSolverInterface, [489](#)
 - Ipopt::Ma57TSolverInterface, [499](#)
 - Ipopt::Ma77SolverInterface, [511](#)
 - Ipopt::Ma86SolverInterface, [520](#)
 - Ipopt::Ma97SolverInterface, [531](#)
 - Ipopt::MumpsSolverInterface, [565](#)
 - Ipopt::WsmSolverInterface, [911](#)
- pivotlmax_
 - Ipopt::Ma27TSolverInterface, [489](#)
 - Ipopt::Ma57TSolverInterface, [499](#)
 - Ipopt::MumpsSolverInterface, [565](#)
- point_perturbation_radius_
 - Ipopt::EquilibrationScaling, [225](#)
 - Ipopt::TNLPAdapter, [847](#)
- PointPerturber
 - Ipopt::PointPerturber, [660](#)
- pool_size
 - ma86_control_d, [513](#)
 - ma86_info_d, [515](#)
- PosTriplet
 - Ipopt::TripletToCSRConverter::TripletEntry, [866](#)
- PrepareAmplForSuffixes
 - Ipopt::AmplSuffixHandler, [77](#)
- PrepareRestoPhaseStart
 - Ipopt::BacktrackingLSAcceptor, [109](#)
 - Ipopt::CGPenaltyLSAcceptor, [133](#)
 - Ipopt::FilterLSAcceptor, [251](#)
 - Ipopt::InexactLSAcceptor, [301](#)
 - Ipopt::PenaltyLSAcceptor, [651](#)
- primal_frac_to_the_bound
 - Ipopt::IpoptCalculatedQuantities, [362](#)
- primal_frac_to_the_bound_cache_
 - Ipopt::IpoptCalculatedQuantities, [372](#)
- PrimalStepSize
 - Ipopt::CGPenaltyData, [126](#)
- Print
 - Ipopt::Filter, [244](#)
 - Ipopt::Journal, [439](#)
 - Ipopt::Matrix, [537](#)
 - Ipopt::PiecewisePenalty, [657](#)
 - Ipopt::Vector, [895](#)
- print_copyright_message
 - Ipopt::IpoptAlgorithm, [336](#)
- print_frequency_iter_
 - Ipopt::OrigIterationOutput, [623](#)
 - Ipopt::RestolIterationOutput, [718](#)
- print_frequency_time_
 - Ipopt::OrigIterationOutput, [623](#)
 - Ipopt::RestolIterationOutput, [718](#)
- print_info_string_
 - Ipopt::OrigIterationOutput, [623](#)
 - Ipopt::RestolIterationOutput, [717](#)
- print_level
 - IpStdCInterface.h, [1005](#)
 - ma77_control_d, [501](#)
 - ma97_control_d, [522](#)
 - mc68_control, [544](#)
- print_levels_
 - Ipopt::Journal, [440](#)
- PrintAllTimingStatistics
 - Ipopt::TimingStatistics, [824](#)
- PrintCopyrightMessage
 - Ipopt::IpoptApplication, [343](#)
- PrintImpl
 - Ipopt::CompoundMatrix, [155](#)
 - Ipopt::CompoundSymMatrix, [164](#)
 - Ipopt::CompoundVector, [175](#)
 - Ipopt::DenseGenMatrix, [193](#)
 - Ipopt::DenseSymMatrix, [198](#)
 - Ipopt::DenseVector, [208](#)
 - Ipopt::DiagMatrix, [219](#)
 - Ipopt::ExpandedMultiVectorMatrix, [231](#)
 - Ipopt::ExpansionMatrix, [237](#)
 - Ipopt::FileJournal, [241](#)
 - Ipopt::GenTMatrix, [268](#)
 - Ipopt::IdentityMatrix, [279](#)
 - Ipopt::Journal, [439](#)
 - Ipopt::LowRankUpdateSymMatrix, [480](#)
 - Ipopt::Matrix, [538](#)
 - Ipopt::MultiVectorMatrix, [557](#)
 - Ipopt::ScaledMatrix, [728](#)
 - Ipopt::StreamJournal, [777](#)
 - Ipopt::SumMatrix, [784](#)
 - Ipopt::SumSymMatrix, [790](#)
 - Ipopt::SymScaledMatrix, [803](#)
 - Ipopt::SymTMatrix, [810](#)
 - Ipopt::TransposeMatrix, [861](#)
 - Ipopt::Vector, [897](#)
 - Ipopt::ZeroMatrix, [914](#)
 - Ipopt::ZeroSymMatrix, [919](#)
- PrintImplOffset
 - Ipopt::DenseVector, [208](#)
 - Ipopt::ExpansionMatrix, [237](#)
 - Ipopt::GenTMatrix, [268](#)
- PrintLatex
 - Ipopt::AmplOptionsList, [74](#)
- PrintList
 - Ipopt::OptionsList, [601](#)
- PrintProblemStatistics
 - Ipopt::IpoptAlgorithm, [336](#)
 - Ipopt::TimingStatistics, [825](#)
- PrintProblemStatistics_
 - Ipopt::TimingStatistics, [827](#)
- PrintStringOverLines
 - Ipopt::Journalist, [442](#)
- PrintTimingStatistics

- Ipopt::OrigIpoptNLP, 615
- PrintUserOptions
 - Ipopt::OptionsList, 601
- printed_num_threads_
 - Ipopt::WsmSolverInterface, 911
- Printf
 - Ipopt::Journal, 439
 - Ipopt::Journalist, 442
- PrintfImpl
 - Ipopt::FileJournal, 241
 - Ipopt::Journal, 439
 - Ipopt::StreamJournal, 777
- PrintfIndented
 - Ipopt::Journalist, 442
- PrivatInfo
 - Ipopt::AmplOptionsList::PrivatInfo, 661
- ProbingMuOracle
 - Ipopt::ProbingMuOracle, 663
- Problem_Source
 - Ipopt::AmplSuffixHandler, 76
- process_target_mu
 - Ipopt::WarmStartIterateInitializer, 903
- ProcessNotification
 - Ipopt::Observer, 592
- ProcessOptions
 - Ipopt::NLP, 572
 - Ipopt::NLPBoundsRemover, 578
 - Ipopt::TNLPAdapter, 842
- ProduceOutput
 - Ipopt::Journalist, 442
- ProvidesDegeneracyDetection
 - Ipopt::Ma77SolverInterface, 511
 - Ipopt::Ma86SolverInterface, 520
 - Ipopt::Ma97SolverInterface, 530
 - Ipopt::MumpsSolverInterface, 564
 - Ipopt::SparseSymLinearSolverInterface, 752
 - Ipopt::TSymLinearSolver, 881
 - Ipopt::WsmSolverInterface, 909
- ProvidesInertia
 - Ipopt::AugRestoSystemSolver, 90
 - Ipopt::AugSystemSolver, 95
 - Ipopt::GenAugSystemSolver, 259
 - Ipopt::GenKKTsSolverInterface, 264
 - Ipopt::IterativePardisoSolverInterface, 422
 - Ipopt::IterativeWsmSolverInterface, 434
 - Ipopt::LowRankAugSystemSolver, 466
 - Ipopt::LowRankSSAugSystemSolver, 473
 - Ipopt::Ma27TSolverInterface, 488
 - Ipopt::Ma57TSolverInterface, 497
 - Ipopt::Ma77SolverInterface, 511
 - Ipopt::Ma86SolverInterface, 519
 - Ipopt::Ma97SolverInterface, 530
 - Ipopt::MumpsSolverInterface, 564
 - Ipopt::PardisoSolverInterface, 628
 - Ipopt::SparseSymLinearSolverInterface, 752
 - Ipopt::StdAugSystemSolver, 763
 - Ipopt::SymLinearSolver, 795
 - Ipopt::TSymLinearSolver, 881
 - Ipopt::WsmSolverInterface, 908
- ptr_
 - Ipopt::SmartPtr, 741
- push_variables
 - Ipopt::DefaultIterateInitializer, 185
- PutValuesInVector
 - Ipopt::TripletHelper, 868
- Px_L
 - Ipopt::IpoptNLP, 396
 - Ipopt::OrigIpoptNLP, 612
 - Ipopt::RestIpoptNLP, 706
- Px_L_
 - Ipopt::OrigIpoptNLP, 618
 - Ipopt::RestIpoptNLP, 710
- Px_U
 - Ipopt::IpoptNLP, 396
 - Ipopt::OrigIpoptNLP, 612
 - Ipopt::RestIpoptNLP, 706
- Px_U_
 - Ipopt::OrigIpoptNLP, 618
 - Ipopt::RestIpoptNLP, 710
- Px_I_orig_
 - Ipopt::NLPBoundsRemover, 580
- px_I_space_
 - Ipopt::OrigIpoptNLP, 616
 - Ipopt::RestIpoptNLP, 709
 - Ipopt::TNLPAdapter, 848
- Px_u_orig_
 - Ipopt::NLPBoundsRemover, 580
- px_u_space_
 - Ipopt::OrigIpoptNLP, 616
 - Ipopt::RestIpoptNLP, 710
 - Ipopt::TNLPAdapter, 848
- Q
 - quality_function_balancing_term_
 - Ipopt::QualityFunctionMuOracle, 670
 - quality_function_centrality_
 - Ipopt::QualityFunctionMuOracle, 670
 - quality_function_max_section_steps_
 - Ipopt::QualityFunctionMuOracle, 670
 - quality_function_norm_
 - Ipopt::QualityFunctionMuOracle, 670
 - quality_function_pd_system
 - Ipopt::AdaptiveMuUpdate, 60
 - quality_function_section_qf_tol_
 - Ipopt::QualityFunctionMuOracle, 670
 - quality_function_section_sigma_tol_
 - Ipopt::QualityFunctionMuOracle, 670
 - QualityFunctionMuOracle

- Ipopt::QualityFunctionMuOracle, 668
- QualityFunctionSearch
 - Ipopt::TimingStatistics, 826
- QualityFunctionSearch_
 - Ipopt::TimingStatistics, 828
- R
- RELAX_BOUNDS
 - Ipopt::TNLPAdapter, 841
- RESTORATION_FAILURE
 - Ipopt, 48
- ReOptimizeNLP
 - Ipopt::IpoptApplication, 342
- ReOptimizeTNLP
 - Ipopt::IpoptApplication, 342
- read_params_dat_
 - Ipopt::IpoptApplication, 344
- ReadFromStream
 - Ipopt::OptionsList, 601
- readnexttoken
 - Ipopt::OptionsList, 601
- recalc_y_
 - Ipopt::IpoptAlgorithm, 337
- recalc_y_feas_tol_
 - Ipopt::IpoptAlgorithm, 338
- RecalcD
 - Ipopt::LimMemQuasiNewtonUpdater, 453
- RecalcL
 - Ipopt::LimMemQuasiNewtonUpdater, 453
- RecalcY
 - Ipopt::LimMemQuasiNewtonUpdater, 453
- RecieveNotification
 - Ipopt::DependentResult, 215
 - Ipopt::Observer, 592
- reduced_diag_
 - Ipopt::LowRankUpdateSymMatrixSpace, 483
- ReducedDiag
 - Ipopt::LowRankUpdateSymMatrix, 480
 - Ipopt::LowRankUpdateSymMatrixSpace, 483
- ReducedInitialize
 - Ipopt::AlgorithmStrategyObject, 68
- ref_point_
 - Ipopt::PointPerturber, 660
- refactorize_
 - Ipopt::Ma27TSolverInterface, 489
 - Ipopt::Ma57TSolverInterface, 499
 - Ipopt::MumpsSolverInterface, 565
- reference_JacC_delta_
 - Ipopt::PenaltyLSAcceptor, 654
- reference_JacD_delta_
 - Ipopt::PenaltyLSAcceptor, 654
- reference_barr_
 - Ipopt::FilterLSAcceptor, 255
 - Ipopt::InexactLSAcceptor, 303
- Ipopt::PenaltyLSAcceptor, 653
 - reference_count_
 - Ipopt::ReferencedObject, 677
 - reference_curr_direct_f_nrm_
 - Ipopt::CGPenaltyLSAcceptor, 138
 - reference_dWd_
 - Ipopt::PenaltyLSAcceptor, 654
 - reference_direct_deriv_penalty_function_
 - Ipopt::CGPenaltyLSAcceptor, 138
 - reference_gradBarrTDelta_
 - Ipopt::FilterLSAcceptor, 255
 - Ipopt::PenaltyLSAcceptor, 653
 - reference_infeasibility_
 - Ipopt::CGPenaltyCq, 121
 - reference_penalty_function_
 - Ipopt::CGPenaltyLSAcceptor, 137
 - reference_pred_
 - Ipopt::InexactLSAcceptor, 304
 - Ipopt::PenaltyLSAcceptor, 654
 - reference_theta_
 - Ipopt::CGPenaltyLSAcceptor, 136
 - Ipopt::FilterLSAcceptor, 255
 - Ipopt::InexactLSAcceptor, 303
 - Ipopt::PenaltyLSAcceptor, 653
- ReferenceCount
 - Ipopt::ReferencedObject, 676
- ReferencedObject
 - Ipopt::ReferencedObject, 676
- refs_red_fact_
 - Ipopt::AdaptiveMuUpdate, 62
- refs_vals_
 - Ipopt::AdaptiveMuUpdate, 62
- reg_options_
 - Ipopt::IpoptApplication, 344
 - Ipopt::OptionsList, 602
- RegOptions
 - Ipopt::IpoptApplication, 342
- RegOptionsList
 - Ipopt::RegisteredOptions, 689
- RegisterAllIpoptOptions
 - Ipopt::IpoptApplication, 343
- RegisterOptions
 - Ipopt::AdaptiveMuUpdate, 59
 - Ipopt::AlgorithmBuilder, 65
 - Ipopt::BacktrackingLineSearch, 101
 - Ipopt::BacktrackingLSAcceptor, 111
 - Ipopt::CGPenaltyCq, 120
 - Ipopt::CGPenaltyLSAcceptor, 134
 - Ipopt::CGPerturbationHandler, 143
 - Ipopt::CGSearchDirCalculator, 149
 - Ipopt::DefaultIterateInitializer, 186
 - Ipopt::EquilibrationScaling, 225
 - Ipopt::FilterLSAcceptor, 252
 - Ipopt::GradientScaling, 274

- Ipopt::InexactAlgorithmBuilder, 283
- Ipopt::InexactCq, 286
- Ipopt::InexactDoglegNormalStep, 296
- Ipopt::InexactLSAcceptor, 302
- Ipopt::InexactNewtonNormalStep, 307
- Ipopt::InexactNormalTerminationTester, 312
- Ipopt::InexactPDSolver, 315
- Ipopt::InexactPDTerminationTester, 320
- Ipopt::InexactSearchDirCalculator, 326
- Ipopt::IpoptAlgorithm, 335
- Ipopt::IpoptApplication, 343
- Ipopt::IpoptCalculatedQuantities, 364
- Ipopt::IpoptData, 385
- Ipopt::IterativePardisoSolverInterface, 422
- Ipopt::IterativeWsmvSolverInterface, 434
- Ipopt::LimMemQuasiNewtonUpdater, 451
- Ipopt::Ma27TSolverInterface, 488
- Ipopt::Ma28TDependencyDetector, 493
- Ipopt::Ma57TSolverInterface, 498
- Ipopt::Ma77SolverInterface, 509
- Ipopt::Ma86SolverInterface, 518
- Ipopt::Ma97SolverInterface, 529
- Ipopt::MinC_1NrmRestorationPhase, 548
- Ipopt::MonotoneMuUpdate, 552
- Ipopt::MumpsSolverInterface, 564
- Ipopt::OptimalityErrorConvergenceCheck, 595
- Ipopt::OrigIpoptNLP, 614
- Ipopt::OrigIterationOutput, 623
- Ipopt::PardisoSolverInterface, 628
- Ipopt::PDFullSpaceSolver, 633
- Ipopt::PDPerturbationHandler, 640
- Ipopt::PDSearchDirCalculator, 645
- Ipopt::PenaltyLSAcceptor, 652
- Ipopt::ProbingMuOracle, 664
- Ipopt::QualityFunctionMuOracle, 669
- Ipopt::RestoConvergenceCheck, 695
- Ipopt::RestoFilterConvergenceCheck, 698
- Ipopt::RestoIpoptNLP, 709
- Ipopt::RestoIterateInitializer, 714
- Ipopt::RestoPenaltyConvergenceCheck, 720
- Ipopt::StandardScalingBase, 758
- Ipopt::TNLPAdapter, 844
- Ipopt::TSymDependencyDetector, 876
- Ipopt::TSymLinearSolver, 881
- Ipopt::WarmStartIterateInitializer, 903
- Ipopt::WsmvSolverInterface, 909
- RegisterOptions_Algorithm
 - Ipopt, 49
- RegisterOptions_CGPenalty
 - Ipopt, 53
- RegisterOptions_Inexact
 - Ipopt, 49
- RegisterOptions_Interfaces
 - Ipopt, 53
- RegisterOptions_LinearSolvers
 - Ipopt, 50
- registered_options_
 - Ipopt::RegisteredOptions, 693
- RegisteredOption
 - Ipopt::RegisteredOption, 681
- RegisteredOptionType
 - Ipopt, 47
- RegisteredOptions
 - Ipopt::RegisteredOptions, 689
- RegisteredOptionsList
 - Ipopt::RegisteredOptions, 692
- registering_category_
 - Ipopt::RegisteredOption, 686
- RegisteringCategory
 - Ipopt::RegisteredOption, 682
 - Ipopt::RegisteredOptions, 690
- RegularMode
 - IpReturnCodes_inc.h, 998
 - Ipopt, 49
- relax_bounds
 - Ipopt::OrigIpoptNLP, 615
- ReleaseInternalDataBackup
 - Ipopt::LimMemQuasiNewtonUpdater, 454
- ReleasePointer_
 - Ipopt::SmartPtr, 740
- ReleaseRef
 - Ipopt::ReferencedObject, 676
- RememberCurrentPointAsAccepted
 - Ipopt::AdaptiveMuUpdate, 59
- replace_bounds_
 - Ipopt::IpoptApplication, 345
- ReportException
 - Ipopt::IpoptException, 391
- RequestAttach
 - Ipopt::Observer, 592
- RequestDetach
 - Ipopt::Observer, 592
- requires_scaling_
 - Ipopt::InexactNormalTerminationTester, 313
 - Ipopt::InexactPDTerminationTester, 321
- rescale_
 - Ipopt::Ma97SolverInterface, 531
- Reset
 - Ipopt::BacktrackingLineSearch, 101
 - Ipopt::BacktrackingLSAcceptor, 109
 - Ipopt::CGPenaltyLSAcceptor, 132
 - Ipopt::FilterLSAcceptor, 251
 - Ipopt::InexactLSAcceptor, 300
 - Ipopt::LineSearch, 461
 - Ipopt::PenaltyLSAcceptor, 651
 - Ipopt::TimedTask, 820
- reset_last_
 - Ipopt::CGPerturbationHandler, 146

- Ipopt::PDPerturbationHandler, 642
- reset_piecewise_penalty_
 - Ipopt::CGPenaltyLSAcceptor, 138
- ResetAdjustedTrialSlacks
 - Ipopt::IpoptCalculatedQuantities, 356
- ResetInfo
 - Ipopt::IpoptData, 385
- ResetList
 - Ipopt::PiecewisePenalty, 657
- ResetSlacks
 - Ipopt::InexactLSAcceptor, 302
- ResetTimes
 - Ipopt::OrigIpoptNLP, 615
 - Ipopt::TimingStatistics, 824
- residual_improvement_factor_
 - Ipopt::PDFullSpaceSolver, 635
- residual_ratio_max_
 - Ipopt::PDFullSpaceSolver, 635
- residual_ratio_singular_
 - Ipopt::PDFullSpaceSolver, 635
- ResortBnds
 - Ipopt::TNLPAdapter, 845
- ResortG
 - Ipopt::TNLPAdapter, 845
- ResortX
 - Ipopt::TNLPAdapter, 845
- resto_alg_
 - Ipopt::MinC_1NrmRestorationPhase, 549
- resto_eq_mult_calculator_
 - Ipopt::RestoIterateInitializer, 715
- resto_failure_feasibility_threshold_
 - Ipopt::MinC_1NrmRestorationPhase, 549
- resto_options_
 - Ipopt::MinC_1NrmRestorationPhase, 549
- resto_orig_iteration_output_
 - Ipopt::RestoIterationOutput, 717
- resto_phase_
 - Ipopt::BacktrackingLineSearch, 106
- resto_pred_
 - Ipopt::InexactLSAcceptor, 305
 - Ipopt::PenaltyLSAcceptor, 654
- RestoConvergenceCheck
 - Ipopt::RestoConvergenceCheck, 695
- RestoFilterConvergenceCheck
 - Ipopt::RestoFilterConvergenceCheck, 698
- RestoIpoptNLP
 - Ipopt::RestoIpoptNLP, 704
- RestoIterateInitializer
 - Ipopt::RestoIterateInitializer, 714
- RestoIterationOutput
 - Ipopt::RestoIterationOutput, 717
- RestoPenaltyConvergenceCheck
 - Ipopt::RestoPenaltyConvergenceCheck, 719
- RestoRestorationPhase
 - Ipopt::RestoRestorationPhase, 724
- restor_counter
 - Ipopt::CGPenaltyData, 126
- restor_counter_
 - Ipopt::CGPenaltyData, 128
- restor_iter
 - Ipopt::CGPenaltyData, 126
- restor_iter_
 - Ipopt::CGPenaltyData, 128
- Restoration_Failed
 - IpReturnCodes_inc.h, 998
 - Ipopt, 49
- RestorationPhaseMode
 - IpReturnCodes_inc.h, 998
 - Ipopt, 49
- RestorationPhase
 - Ipopt::RestorationPhase, 722
- restore_accepted_iterate_
 - Ipopt::AdaptiveMuUpdate, 62
- RestoreAcceptablePoint
 - Ipopt::BacktrackingLineSearch, 103
- RestoreBestPoint
 - Ipopt::CGPenaltyLSAcceptor, 135
- RestoreInternalDataBackup
 - Ipopt::LimMemQuasiNewtonUpdater, 454
- RestoredIterate
 - Ipopt::BacktrackingLSAcceptor, 111
 - Ipopt::CGPenaltyLSAcceptor, 134
- result_
 - Ipopt::DependentResult, 216
- rethrow_nonipoptexception_
 - Ipopt::IpoptApplication, 344
- RethrowNonIpoptException
 - Ipopt::IpoptApplication, 343
- Rho
 - Ipopt::RestoIpoptNLP, 708
- rho_
 - Ipopt::InexactLSAcceptor, 303
 - Ipopt::InexactPDTerminationTester, 321
 - Ipopt::PenaltyLSAcceptor, 653
 - Ipopt::RestoIpoptNLP, 711
- Rhs_cR
 - Ipopt::AugRestoSystemSolver, 91
- rhs_cR_cache_
 - Ipopt::AugRestoSystemSolver, 92
- Rhs_dR
 - Ipopt::AugRestoSystemSolver, 91
- rhs_dR_cache_
 - Ipopt::AugRestoSystemSolver, 92
- rigorous_
 - Ipopt::BacktrackingLineSearch, 106
- row_full_thresh
 - mc68_control, 544
- row_scaling_

- Ipopt::ScaledMatrixSpace, [731](#)
- row_search
 - mc68_control, [544](#)
- RowColScaling
 - Ipopt::SymScaledMatrix, [802](#)
 - Ipopt::SymScaledMatrixSpace, [806](#)
- RowScaling
 - Ipopt::ScaledMatrix, [727](#)
 - Ipopt::ScaledMatrixSpace, [731](#)
- RowVectorSpace
 - Ipopt::ExpandedMultiVectorMatrix, [230](#)
 - Ipopt::ExpandedMultiVectorMatrixSpace, [233](#)
- rspace
 - ma77_control_d, [503](#)
 - ma77_info_d, [506](#)
 - ma97_control_d, [523](#)
 - ma97_info, [525](#)
- S
- s
 - Ipopt::IteratesVector, [406](#)
- SAFE_MIN_DUAL_INFEAS_ALPHA_FOR_Y
 - Ipopt::BacktrackingLineSearch, [100](#)
- SCALAR1
 - Ipopt::LimMemQuasiNewtonUpdater, [451](#)
- SCALAR2
 - Ipopt::LimMemQuasiNewtonUpdater, [451](#)
- SCALAR3
 - Ipopt::LimMemQuasiNewtonUpdater, [451](#)
- SCALAR4
 - Ipopt::LimMemQuasiNewtonUpdater, [451](#)
- SECOND_ORDER_TEST
 - Ipopt::TNLPAdapter, [842](#)
- SR1
 - Ipopt::LimMemQuasiNewtonUpdater, [450](#)
- STOP_AT_ACCEPTABLE_POINT
 - Ipopt, [48](#)
- STOP_AT_TINY_STEP
 - Ipopt, [48](#)
- SUCCESS
 - Ipopt, [48](#)
- SWITCH_AT_START
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_AT_START_REUSE
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_NDELAY
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_NDELAY_REUSE
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_NEVER
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_OD_ND
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_OD_ND_REUSE
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_ON_DEMAND
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_ON_DEMAND_REUSE
 - Ipopt::Ma97SolverInterface, [528](#)
- SWITCH_ONCE
 - Ipopt::InexactSearchDirCalculator, [325](#)
- SYMSOLVER_CALL_AGAIN
 - Ipopt, [46](#)
- SYMSOLVER_FATAL_ERROR
 - Ipopt, [46](#)
- SYMSOLVER_SINGULAR
 - Ipopt, [46](#)
- SYMSOLVER_SUCCESS
 - Ipopt, [46](#)
- SYMSOLVER_WRONG_INERTIA
 - Ipopt, [46](#)
- S_
 - Ipopt::LimMemQuasiNewtonUpdater, [456](#)
- s_NonConst
 - Ipopt::IteratesVector, [406](#)
- s_max_
 - Ipopt::IpoptCalculatedQuantities, [365](#)
- S_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [457](#)
- s_phi_
 - Ipopt::FilterLSAcceptor, [253](#)
- s_space_
 - Ipopt::IteratesVectorSpace, [414](#)
- s_theta_
 - Ipopt::FilterLSAcceptor, [253](#)
- STDRS_
 - Ipopt::LimMemQuasiNewtonUpdater, [457](#)
- STDRS_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [459](#)
- Scal
 - Ipopt::Vector, [892](#)
- Scallmpl
 - Ipopt::CompoundVector, [172](#)
 - Ipopt::DenseVector, [205](#)
 - Ipopt::Vector, [895](#)
- Scalar
 - Ipopt::DenseVector, [205](#)
- scalar_
 - Ipopt::DenseVector, [209](#)
- scalar_dependents_
 - Ipopt::DependentResult, [216](#)
- scale_opts
 - Ipopt::Ma97SolverInterface, [528](#)
- ScaleColumns
 - Ipopt::DenseGenMatrix, [191](#)
 - Ipopt::MultiVectorMatrix, [556](#)
- ScaleNameToNum
 - Ipopt::Ma97SolverInterface, [530](#)

- ScaleRows
 - Ipopt::MultiVectorMatrix, [556](#)
- ScaleSigma
 - Ipopt::QualityFunctionMuOracle, [669](#)
- scaled_compl_
 - Ipopt::SolveStatistics, [746](#)
- scaled_constr_viol_
 - Ipopt::SolveStatistics, [746](#)
- scaled_dual_inf_
 - Ipopt::SolveStatistics, [746](#)
- scaled_h_space_
 - Ipopt::OrigIpoptNLP, [617](#)
 - Ipopt::StandardScalingBase, [759](#)
- scaled_jac_c_space_
 - Ipopt::OrigIpoptNLP, [617](#)
 - Ipopt::StandardScalingBase, [759](#)
- scaled_jac_d_space_
 - Ipopt::OrigIpoptNLP, [617](#)
 - Ipopt::StandardScalingBase, [759](#)
- scaled_kkt_error_
 - Ipopt::SolveStatistics, [747](#)
- scaled_obj_val_
 - Ipopt::SolveStatistics, [746](#)
- ScaledInfeasibilities
 - Ipopt::SolveStatistics, [745](#)
- ScaledMatrix
 - Ipopt::ScaledMatrix, [726](#)
- ScaledMatrixSpace
 - Ipopt::ScaledMatrixSpace, [730](#)
- scaling
 - ma86_control_d, [513](#)
 - ma97_control_d, [522](#)
- scaling_
 - Ipopt::Ma97SolverInterface, [531](#)
 - Ipopt::SymScaledMatrixSpace, [806](#)
- scaling_constr_target_gradient_
 - Ipopt::GradientScaling, [274](#)
- scaling_factors_
 - Ipopt::TSymLinearSolver, [882](#)
- scaling_max_gradient_
 - Ipopt::GradientScaling, [274](#)
- scaling_method_
 - Ipopt::TSymLinearSolver, [882](#)
- scaling_min_value_
 - Ipopt::GradientScaling, [274](#)
- scaling_obj_target_gradient_
 - Ipopt::GradientScaling, [274](#)
- scaling_type_
 - Ipopt::Ma97SolverInterface, [531](#)
- scaling_val_
 - Ipopt::Ma97SolverInterface, [532](#)
- SdotS_
 - Ipopt::LimMemQuasiNewtonUpdater, [457](#)
- SdotS_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [458](#)
- SdotS_uptodate_
 - Ipopt::LimMemQuasiNewtonUpdater, [457](#)
- SdotS_uptodate_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [458](#)
- Search_Direction_Becomes_Too_Small
 - IpReturnCodes_inc.h, [997](#)
 - Ipopt, [48](#)
- search_dir_calculator_
 - Ipopt::IpoptAlgorithm, [337](#)
- SearchDirCalc
 - Ipopt::IpoptAlgorithm, [335](#)
- SearchDirectionCalculator
 - Ipopt::SearchDirectionCalculator, [733](#)
- Set
 - Ipopt::TripletToCSRConverter::TripletEntry, [865](#)
 - Ipopt::Vector, [892](#)
- Set_W
 - Ipopt::IpoptData, [381](#)
- set_active_objective
 - Ipopt::AmplTNLP, [84](#)
- set_active_objective_called_
 - Ipopt::AmplTNLP, [86](#)
- Set_bound_mult
 - Ipopt::IteratesVector, [411](#)
- Set_bound_mult_NonConst
 - Ipopt::IteratesVector, [411](#)
- Set_c_Avc_norm_cauchy
 - Ipopt::InexactNormalTerminationTester, [312](#)
- set_compute_normal
 - Ipopt::InexactData, [293](#)
- set_curr_nu
 - Ipopt::InexactData, [292](#)
- set_delta
 - Ipopt::IpoptData, [380](#), [381](#)
- set_delta_aff
 - Ipopt::IpoptData, [381](#)
- set_delta_cgfast
 - Ipopt::CGPenaltyData, [125](#)
- set_delta_cgpen
 - Ipopt::CGPenaltyData, [125](#)
- Set_eq_mult
 - Ipopt::IteratesVector, [410](#)
- Set_eq_mult_NonConst
 - Ipopt::IteratesVector, [410](#)
- set_full_step_accepted
 - Ipopt::InexactData, [292](#)
- Set_info_alpha_dual
 - Ipopt::IpoptData, [384](#)
- Set_info_alpha_primal
 - Ipopt::IpoptData, [383](#)
- Set_info_alpha_primal_char
 - Ipopt::IpoptData, [384](#)
- Set_info_iters_since_header

Ipopt::IpoptData, [385](#)
 Set_info_last_output
 Ipopt::IpoptData, [384](#)
 Set_info_ls_count
 Ipopt::IpoptData, [384](#)
 Set_info_regu_x
 Ipopt::IpoptData, [383](#)
 Set_info_skip_output
 Ipopt::IpoptData, [384](#)
 set_integer_metadata_for_con
 Ipopt::AmplTNLP, [84](#)
 set_integer_metadata_for_var
 Ipopt::AmplTNLP, [84](#)
 Set_iter_count
 Ipopt::IpoptData, [382](#)
 Set_kkt_penalty
 Ipopt::CGPenaltyData, [127](#)
 Set_mu
 Ipopt::IpoptData, [382](#)
 set_next_compute_normal
 Ipopt::InexactData, [293](#)
 set_normal_s
 Ipopt::InexactData, [292](#)
 set_normal_x
 Ipopt::InexactData, [292](#)
 set_numeric_metadata_for_con
 Ipopt::AmplTNLP, [84](#)
 set_numeric_metadata_for_var
 Ipopt::AmplTNLP, [84](#)
 Set_penalty
 Ipopt::CGPenaltyData, [126](#)
 Set_primal
 Ipopt::IteratesVector, [410](#)
 Set_primal_NonConst
 Ipopt::IteratesVector, [410](#)
 Set_s
 Ipopt::IteratesVector, [406](#)
 Set_s_NonConst
 Ipopt::IteratesVector, [406](#)
 set_string_metadata_for_con
 Ipopt::AmplTNLP, [84](#)
 set_string_metadata_for_var
 Ipopt::AmplTNLP, [84](#)
 set_tangential_s
 Ipopt::InexactData, [292](#)
 set_tangential_x
 Ipopt::InexactData, [292](#)
 Set_tau
 Ipopt::IpoptData, [382](#)
 Set_tiny_step_flag
 Ipopt::IpoptData, [383](#)
 Set_tol
 Ipopt::IpoptData, [383](#)
 set_trial

Ipopt::IpoptData, [380](#)
 Set_v_L
 Ipopt::IteratesVector, [409](#)
 Set_v_L_NonConst
 Ipopt::IteratesVector, [409](#)
 Set_v_U
 Ipopt::IteratesVector, [410](#)
 Set_v_U_NonConst
 Ipopt::IteratesVector, [410](#)
 set_values_from_scalar
 Ipopt::DenseVector, [208](#)
 Set_x
 Ipopt::IteratesVector, [405](#)
 Set_x_NonConst
 Ipopt::IteratesVector, [405](#)
 Set_y_c
 Ipopt::IteratesVector, [407](#)
 Set_y_c_NonConst
 Ipopt::IteratesVector, [407](#)
 Set_y_d
 Ipopt::IteratesVector, [407](#)
 Set_y_d_NonConst
 Ipopt::IteratesVector, [407](#)
 Set_z_L
 Ipopt::IteratesVector, [408](#)
 Set_z_L_NonConst
 Ipopt::IteratesVector, [408](#)
 Set_z_U
 Ipopt::IteratesVector, [409](#)
 Set_z_U_NonConst
 Ipopt::IteratesVector, [409](#)
 SetAddCq
 Ipopt::IpoptCalculatedQuantities, [355](#)
 SetAddData
 Ipopt::IpoptData, [385](#)
 SetAllPrintLevels
 Ipopt::Journal, [439](#)
 SetBlockCols
 Ipopt::CompoundMatrixSpace, [158](#)
 SetBlockDim
 Ipopt::CompoundSymMatrixSpace, [167](#)
 SetBlockRows
 Ipopt::CompoundMatrixSpace, [158](#)
 SetComp
 Ipopt::CompoundMatrix, [153](#)
 Ipopt::CompoundSymMatrix, [163](#)
 Ipopt::CompoundVector, [172](#)
 SetCompNonConst
 Ipopt::CompoundMatrix, [153](#)
 Ipopt::CompoundSymMatrix, [163](#)
 Ipopt::CompoundVector, [172](#)
 SetCompSpace
 Ipopt::CompoundMatrixSpace, [158](#)
 Ipopt::CompoundSymMatrixSpace, [167](#)

- [Ipopt::CompoundVectorSpace](#), 178
 - [Ipopt::IteratesVectorSpace](#), 414
- [SetCurrPenaltyPert](#)
 - [Ipopt::CGPenaltyData](#), 126
- [SetDefaultInteger](#)
 - [Ipopt::RegisteredOption](#), 684
- [SetDefaultNumber](#)
 - [Ipopt::RegisteredOption](#), 684
- [SetDefaultString](#)
 - [Ipopt::RegisteredOption](#), 684
- [SetDiag](#)
 - [Ipopt::DiagMatrix](#), 218
 - [Ipopt::LowRankUpdateSymMatrix](#), 479
- [SetFactor](#)
 - [Ipopt::IdentityMatrix](#), 278
- [SetFreeMuMode](#)
 - [Ipopt::IpoptData](#), 382
- [SetFromRawPtr_](#)
 - [Ipopt::SmartPtr](#), 740
- [SetFromSmartPtr_](#)
 - [Ipopt::SmartPtr](#), 740
- [SetHaveAffineDeltas](#)
 - [Ipopt::IpoptData](#), 382
- [SetHaveCgFastDeltas](#)
 - [Ipopt::CGPenaltyData](#), 126
- [SetHaveCgPenDeltas](#)
 - [Ipopt::CGPenaltyData](#), 125
- [SetHaveDeltas](#)
 - [Ipopt::IpoptData](#), 381
- [SetImpl](#)
 - [Ipopt::CompoundVector](#), 173
 - [Ipopt::DenseVector](#), 206
 - [Ipopt::Vector](#), 895
- [SetInitialIterates](#)
 - [Ipopt::DefaultIterateInitializer](#), 185
 - [Ipopt::IterateInitializer](#), 400
 - [Ipopt::RestoIterateInitializer](#), 714
 - [Ipopt::WarmStartIterateInitializer](#), 903
- [SetIntegerMetaData](#)
 - [Ipopt::DenseVectorSpace](#), 212
- [SetIntegerValue](#)
 - [Ipopt::OptionsList](#), 600
- [SetIntegerValueIfUnset](#)
 - [Ipopt::OptionsList](#), 601
- [SetJournalist](#)
 - [Ipopt::OptionsList](#), 600
- [SetKKTPenaltyUninitialized](#)
 - [Ipopt::CGPenaltyData](#), 127
- [SetLongDescription](#)
 - [Ipopt::RegisteredOption](#), 682
- [SetLowerInteger](#)
 - [Ipopt::RegisteredOption](#), 683
- [SetLowerNumber](#)
 - [Ipopt::RegisteredOption](#), 683
- [SetName](#)
 - [Ipopt::RegisteredOption](#), 681
- [SetNeverTryPureNewton](#)
 - [Ipopt::CGPenaltyData](#), 126
- [SetNumericMetaData](#)
 - [Ipopt::DenseVectorSpace](#), 212
- [SetNumericValue](#)
 - [Ipopt::OptionsList](#), 600
- [SetNumericValueIfUnset](#)
 - [Ipopt::OptionsList](#), 600
- [SetOrigLSAcceptor](#)
 - [Ipopt::RestoConvergenceCheck](#), 695
 - [Ipopt::RestoFilterConvergenceCheck](#), 698
 - [Ipopt::RestoPenaltyConvergenceCheck](#), 720
- [SetOutputStream](#)
 - [Ipopt::StreamJournal](#), 777
- [setPDPert](#)
 - [Ipopt::IpoptData](#), 385
- [SetPenaltyUninitialized](#)
 - [Ipopt::CGPenaltyData](#), 127
- [SetPrimalStepSize](#)
 - [Ipopt::CGPenaltyData](#), 126
- [SetPrintLevel](#)
 - [Ipopt::Journal](#), 438
- [SetRegisteredOptions](#)
 - [Ipopt::OptionsList](#), 600
- [SetRegisteringCategory](#)
 - [Ipopt::RegisteredOption](#), 682
 - [Ipopt::RegisteredOptions](#), 690
- [SetRestorCounter](#)
 - [Ipopt::CGPenaltyData](#), 126
- [SetRestorIter](#)
 - [Ipopt::CGPenaltyData](#), 126
- [SetRigorousLineSearch](#)
 - [Ipopt::BacktrackingLineSearch](#), 101
 - [Ipopt::LineSearch](#), 461
- [SetShortDescription](#)
 - [Ipopt::RegisteredOption](#), 681
- [SetStringMetaData](#)
 - [Ipopt::DenseVectorSpace](#), 212
- [SetStringValue](#)
 - [Ipopt::OptionsList](#), 600
- [SetStringValueIfUnset](#)
 - [Ipopt::OptionsList](#), 600
- [SetTerm](#)
 - [Ipopt::SumMatrix](#), 783
 - [Ipopt::SumSymMatrix](#), 789
- [SetTermSpace](#)
 - [Ipopt::SumMatrixSpace](#), 786
 - [Ipopt::SumSymMatrixSpace](#), 792
- [SetTrialBoundMultipliersFromStep](#)
 - [Ipopt::IpoptData](#), 380
- [SetTrialEqMultipliersFromStep](#)
 - [Ipopt::IpoptData](#), 380

- SetTrialPrimalVariablesFromStep
 - Ipopt::IpoptData, [380](#)
- SetType
 - Ipopt::RegisteredOption, [682](#)
- SetU
 - Ipopt::LowRankUpdateSymMatrix, [479](#)
- SetUnscaledMatrix
 - Ipopt::ScaledMatrix, [727](#)
 - Ipopt::SymScaledMatrix, [802](#)
- SetUnscaledMatrixNonConst
 - Ipopt::ScaledMatrix, [727](#)
 - Ipopt::SymScaledMatrix, [802](#)
- SetUpperInteger
 - Ipopt::RegisteredOption, [683](#)
- SetUpperNumber
 - Ipopt::RegisteredOption, [683](#)
- SetV
 - Ipopt::LowRankUpdateSymMatrix, [479](#)
- SetValues
 - Ipopt::DenseVector, [204](#)
 - Ipopt::GenTMatrix, [267](#)
 - Ipopt::SymTMatrix, [809](#)
- SetVector
 - Ipopt::ExpandedMultiVectorMatrix, [230](#)
 - Ipopt::MultiVectorMatrix, [556](#)
- SetVectorNonConst
 - Ipopt::MultiVectorMatrix, [556](#)
- SetW
 - Ipopt::LimMemQuasiNewtonUpdater, [454](#)
- ShiftDenseVector
 - Ipopt::LimMemQuasiNewtonUpdater, [453](#)
- ShiftLMatrix
 - Ipopt::LimMemQuasiNewtonUpdater, [453](#)
- ShiftMultiVector
 - Ipopt::LimMemQuasiNewtonUpdater, [452](#)
- ShiftSTDRSMatrix
 - Ipopt::LimMemQuasiNewtonUpdater, [453](#)
- ShiftSdotSMatrix
 - Ipopt::LimMemQuasiNewtonUpdater, [453](#)
- short_description_
 - Ipopt::RegisteredOption, [685](#)
- ShortDescription
 - Ipopt::RegisteredOption, [681](#)
- sigma_
 - Ipopt::LimMemQuasiNewtonUpdater, [456](#)
- sigma_max_
 - Ipopt::ProbingMuOracle, [664](#)
 - Ipopt::QualityFunctionMuOracle, [670](#)
- sigma_min_
 - Ipopt::QualityFunctionMuOracle, [670](#)
- sigma_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [458](#)
- sigma_safe_max_
 - Ipopt::LimMemQuasiNewtonUpdater, [455](#)
- sigma_safe_min_
 - Ipopt::LimMemQuasiNewtonUpdater, [455](#)
- Sigma_tilde_n_c_inv
 - Ipopt::AugRestoSystemSolver, [91](#)
- sigma_tilde_n_c_inv_cache_
 - Ipopt::AugRestoSystemSolver, [92](#)
- Sigma_tilde_n_d_inv
 - Ipopt::AugRestoSystemSolver, [91](#)
- sigma_tilde_n_d_inv_cache_
 - Ipopt::AugRestoSystemSolver, [92](#)
- Sigma_tilde_p_c_inv
 - Ipopt::AugRestoSystemSolver, [91](#)
- sigma_tilde_p_c_inv_cache_
 - Ipopt::AugRestoSystemSolver, [92](#)
- Sigma_tilde_p_d_inv
 - Ipopt::AugRestoSystemSolver, [91](#)
- sigma_tilde_p_d_inv_cache_
 - Ipopt::AugRestoSystemSolver, [92](#)
- SinvBlrmZMTdBr
 - Ipopt::Matrix, [536](#)
- SinvBlrmZMTdBrImpl
 - Ipopt::CompoundMatrix, [154](#)
 - Ipopt::ExpansionMatrix, [236](#)
 - Ipopt::Matrix, [537](#)
 - Ipopt::ScaledMatrix, [728](#)
- SinvBlrmZPTdBr
 - Ipopt::PDFullSpaceSolver, [634](#)
- skip_corr_if_neg_curv_
 - Ipopt::FilterLSAcceptor, [254](#)
- skip_corr_in_monotone_mode_
 - Ipopt::FilterLSAcceptor, [254](#)
- skip_inertia_check_
 - Ipopt::IterativePardisoSolverInterface, [424](#)
 - Ipopt::Ma27TSolverInterface, [490](#)
 - Ipopt::PardisoSolverInterface, [629](#)
 - Ipopt::WsmvSolverInterface, [910](#)
- skip_orig_aug_solver_init_
 - Ipopt::AugRestoSystemSolver, [92](#)
- skip_print_problem_stats_
 - Ipopt::IpoptAlgorithm, [337](#)
- skipped_line_search_
 - Ipopt::BacktrackingLineSearch, [106](#)
- slack_bound_frac_
 - Ipopt::DefaultIterateInitializer, [186](#)
- slack_bound_push_
 - Ipopt::DefaultIterateInitializer, [186](#)
- slack_move_
 - Ipopt::IpoptCalculatedQuantities, [366](#)
- slack_scale_max_
 - Ipopt::InexactCq, [289](#)
- slack_scaled_norm
 - Ipopt::InexactCq, [287](#)
- slack_scaled_norm_cache_
 - Ipopt::InexactCq, [288](#)

- SlackBasedTSymScalingMethod
 - Ipopt::SlackBasedTSymScalingMethod, [734](#), [735](#)
- small
 - ma77_control_d, [502](#)
 - ma97_control_d, [522](#)
- small_
 - ma86_control_d, [513](#)
- SmartPtr
 - Ipopt::SmartPtr, [739](#)
- Snprintf
 - Ipopt, [52](#)
- soHandle_t
 - LibraryHandler.h, [992](#)
- soft_resto_counter_
 - Ipopt::BacktrackingLineSearch, [106](#)
- soft_resto_perror_reduction_factor_
 - Ipopt::BacktrackingLineSearch, [103](#)
- Solve
 - Ipopt::AugRestoSystemSolver, [90](#)
 - Ipopt::AugSystemSolver, [94](#)
 - Ipopt::InexactPDSolver, [315](#)
 - Ipopt::IterativePardisoSolverInterface, [423](#)
 - Ipopt::IterativeWsmvSolverInterface, [435](#)
 - Ipopt::LowRankAugSystemSolver, [466](#)
 - Ipopt::LowRankSSAugSystemSolver, [473](#)
 - Ipopt::MumpsSolverInterface, [564](#)
 - Ipopt::PardisoSolverInterface, [628](#)
 - Ipopt::PDFullSpaceSolver, [633](#)
 - Ipopt::PDSolverInterface, [647](#)
 - Ipopt::SymLinearSolver, [794](#)
 - Ipopt::WsmvSolverInterface, [909](#)
- Solve_Succeeded
 - IpReturnCodes_inc.h, [997](#)
 - Ipopt, [48](#)
- solve_blas3
 - ma97_control_d, [522](#)
- solve_mf
 - ma97_control_d, [522](#)
- solve_min
 - ma97_control_d, [522](#)
- solve_quadratic
 - Ipopt::RestoIterateInitializer, [715](#)
 - Ipopt::RestoRestorationPhase, [724](#)
- SolveMultiVector
 - Ipopt::LowRankAugSystemSolver, [467](#)
- SolveOnce
 - Ipopt::PDFullSpaceSolver, [633](#)
- SolveStatistics
 - Ipopt::SolveStatistics, [744](#)
- Solved_To_Acceptable_Level
 - IpReturnCodes_inc.h, [997](#)
 - Ipopt, [48](#)
- solver_interface_
 - Ipopt::GenAugSystemSolver, [260](#)
 - Ipopt::TSymLinearSolver, [882](#)
- SolverReturn
 - Ipopt, [48](#)
- SparseSymLinearSolverInterface
 - Ipopt::SparseSymLinearSolverInterface, [750](#)
- SpecialAddForLMSR1
 - Ipopt::DenseSymMatrix, [198](#)
- SplitEigenvalues
 - Ipopt::LimMemQuasiNewtonUpdater, [453](#)
- stale_
 - Ipopt::DependentResult, [216](#)
- StandardScalingBase
 - Ipopt::StandardScalingBase, [755](#)
- Start
 - Ipopt::TimedTask, [820](#)
- start_called_
 - Ipopt::TimedTask, [822](#)
- start_cputime_
 - Ipopt::TimedTask, [821](#)
- start_lam_
 - Ipopt::StdInterfaceTNLP, [773](#)
- start_systime_
 - Ipopt::TimedTask, [821](#)
- start_walltime_
 - Ipopt::TimedTask, [822](#)
- start_with_resto_
 - Ipopt::BacktrackingLineSearch, [105](#)
- start_x_
 - Ipopt::StdInterfaceTNLP, [773](#)
- start_z_L_
 - Ipopt::StdInterfaceTNLP, [773](#)
- start_z_U_
 - Ipopt::StdInterfaceTNLP, [773](#)
- StartWatchDog
 - Ipopt::BacktrackingLineSearch, [102](#)
 - Ipopt::BacktrackingLSAcceptor, [110](#)
 - Ipopt::CGPenaltyLSAcceptor, [134](#)
 - Ipopt::FilterLSAcceptor, [252](#)
 - Ipopt::InexactLSAcceptor, [301](#)
 - Ipopt::PenaltyLSAcceptor, [652](#)
- stat
 - ma77_info_d, [505](#)
 - ma86_info_d, [515](#)
 - ma97_info, [525](#)
 - mc68_info, [545](#)
- static_
 - ma77_control_d, [503](#)
 - ma86_control_d, [513](#)
- Statistics
 - Ipopt::IpoptApplication, [343](#)
- statistics_
 - Ipopt::IpoptApplication, [344](#)
- StdAugSystemSolver
 - Ipopt::StdAugSystemSolver, [762](#)

- StdAugSystemSolverMultiSolve
 - Ipopt::TimingStatistics, [825](#)
- StdAugSystemSolverMultiSolve_
 - Ipopt::TimingStatistics, [827](#)
- StdInterfaceTNLP
 - Ipopt::StdInterfaceTNLP, [770](#)
- StopWatchDog
 - Ipopt::BacktrackingLineSearch, [102](#)
 - Ipopt::BacktrackingLSAccepter, [110](#)
 - Ipopt::CGPenaltyLSAccepter, [134](#)
 - Ipopt::FilterLSAccepter, [252](#)
 - Ipopt::InexactLSAccepter, [301](#)
 - Ipopt::PenaltyLSAccepter, [652](#)
- storage
 - ma77_control_d, [502](#)
 - ma77_info_d, [506](#)
- storage_indef
 - ma77_control_d, [503](#)
- StoreAcceptablePoint
 - Ipopt::BacktrackingLineSearch, [103](#)
- StoreBestPoint
 - Ipopt::CGPenaltyLSAccepter, [135](#)
- StoreInternalDataBackup
 - Ipopt::LimMemQuasiNewtonUpdater, [454](#)
- StreamJournal
 - Ipopt::StreamJournal, [776](#)
- String_Option
 - Ipopt::AmplOptionsList, [73](#)
- string_entry
 - Ipopt::RegisteredOption::string_entry, [778](#)
- string_equal_insensitive
 - Ipopt::RegisteredOption, [685](#)
- string_meta_data_
 - Ipopt::DenseVectorSpace, [212](#)
- StringMetaDataMapType
 - Ipopt, [45](#)
 - Ipopt::TNLP, [831](#)
- Subject
 - Ipopt::Observer, [592](#)
 - Ipopt::Subject, [780](#)
- subjects_
 - Ipopt::Observer, [592](#)
- successive_resto_iter_
 - Ipopt::RestoConvergenceCheck, [696](#)
- Suffix_Source
 - Ipopt::AmplSuffixHandler, [76](#)
- Suffix_Type
 - Ipopt::AmplSuffixHandler, [76](#)
- suffix_handler_
 - Ipopt::AmplTNLP, [87](#)
- suffix_ids_
 - Ipopt::AmplSuffixHandler, [77](#)
- suffix_sources_
 - Ipopt::AmplSuffixHandler, [77](#)
- suffix_types_
 - Ipopt::AmplSuffixHandler, [77](#)
- suftab_
 - Ipopt::AmplSuffixHandler, [77](#)
- Sum
 - Ipopt::Vector, [894](#)
- sum_cache_tag_
 - Ipopt::Vector, [898](#)
- SumImpl
 - Ipopt::CompoundVector, [174](#)
 - Ipopt::DenseVector, [207](#)
 - Ipopt::Vector, [897](#)
- SumLogs
 - Ipopt::Vector, [894](#)
- SumLogsImpl
 - Ipopt::CompoundVector, [174](#)
 - Ipopt::DenseVector, [207](#)
 - Ipopt::Vector, [897](#)
- SumMatrix
 - Ipopt::SumMatrix, [783](#)
- SumMatrixSpace
 - Ipopt::SumMatrixSpace, [786](#)
- SumSymMatrix
 - Ipopt::SumSymMatrix, [789](#)
- SumSymMatrixSpace
 - Ipopt::SumSymMatrixSpace, [791](#)
- sumlogs_cache_tag_
 - Ipopt::Vector, [899](#)
- sumsym_space_x_
 - Ipopt::StdAugSystemSolver, [764](#)
- swap
 - Ipopt, [51](#)
- switch_
 - Ipopt::Ma97SolverInterface, [531](#)
- SymLinearSolver
 - Ipopt::SymLinearSolver, [794](#)
- SymMatrix
 - Ipopt::SymMatrix, [796](#)
- SymMatrixSpace
 - Ipopt::SymMatrixSpace, [799](#)
- SymScaledMatrix
 - Ipopt::SymScaledMatrix, [802](#)
- SymScaledMatrixSpace
 - Ipopt::SymScaledMatrixSpace, [805](#)
- SymTMatrix
 - Ipopt::SymTMatrix, [809](#)
 - Ipopt::SymTMatrixSpace, [813](#)
- SymTMatrixSpace
 - Ipopt::SymTMatrixSpace, [812](#)
- SymbolicFactorization
 - Ipopt::IterativePardisoSolverInterface, [423](#)
 - Ipopt::IterativeWsmvSolverInterface, [434](#)
 - Ipopt::Ma27TSolverInterface, [488](#)
 - Ipopt::Ma57TSolverInterface, [498](#)

- Ipopt::MumpsSolverInterface, [564](#)
 - Ipopt::PardisoSolverInterface, [628](#)
 - Ipopt::WsmvSolverInterface, [909](#)
- SysTime
 - Ipopt, [52](#)
- T
- TEST_1_SATISFIED
 - Ipopt::IterativeSolverTerminationTester, [429](#)
- TEST_2_SATISFIED
 - Ipopt::IterativeSolverTerminationTester, [429](#)
- TEST_3_SATISFIED
 - Ipopt::IterativeSolverTerminationTester, [429](#)
- TEST_DELTA_C_EQ_0_DELTA_X_EQ_0
 - Ipopt::CGPerturbationHandler, [142](#)
 - Ipopt::PDPerturbationHandler, [638](#)
- TEST_DELTA_C_EQ_0_DELTA_X_GT_0
 - Ipopt::CGPerturbationHandler, [142](#)
 - Ipopt::PDPerturbationHandler, [638](#)
- TEST_DELTA_C_GT_0_DELTA_X_EQ_0
 - Ipopt::CGPerturbationHandler, [142](#)
 - Ipopt::PDPerturbationHandler, [638](#)
- TEST_DELTA_C_GT_0_DELTA_X_GT_0
 - Ipopt::CGPerturbationHandler, [142](#)
 - Ipopt::PDPerturbationHandler, [639](#)
- TOO_FEW_DEGREES_OF_FREEDOM
 - Ipopt, [48](#)
- TDependencyDetector
 - Ipopt::TDependencyDetector, [818](#)
- THROW_EXCEPTION
 - IpException.hpp, [969](#)
- TNLP
 - Ipopt::TNLP, [832](#)
- TNLPAAdapter
 - Ipopt::TNLPAdapter, [842](#)
- TNLPReducer
 - Ipopt::TNLPReducer, [854](#)
- TRUE
 - IpStdCInterface.h, [1001](#)
- TSymDependencyDetector
 - Ipopt::TSymDependencyDetector, [876](#)
- TSymLinearSolver
 - Ipopt::TSymLinearSolver, [880](#)
- TSymScalingMethod
 - Ipopt::TSymScalingMethod, [884](#)
- Tag
 - Ipopt::TaggedObject, [816](#)
- tagcount_
 - Ipopt::TaggedObject, [816](#)
- TaggedObject
 - Ipopt::TaggedObject, [816](#)
- tangential_s
 - Ipopt::InexactData, [292](#)
- tangential_s_
 - Ipopt::InexactData, [293](#)
- tangential_x
 - Ipopt::InexactData, [292](#)
- tangential_x_
 - Ipopt::InexactData, [293](#)
- Task1
 - Ipopt::TimingStatistics, [826](#)
- Task1_
 - Ipopt::TimingStatistics, [828](#)
- Task2
 - Ipopt::TimingStatistics, [826](#)
- Task2_
 - Ipopt::TimingStatistics, [828](#)
- Task3
 - Ipopt::TimingStatistics, [826](#)
- Task3_
 - Ipopt::TimingStatistics, [828](#)
- Task4
 - Ipopt::TimingStatistics, [826](#)
- Task4_
 - Ipopt::TimingStatistics, [828](#)
- Task5
 - Ipopt::TimingStatistics, [826](#)
- Task5_
 - Ipopt::TimingStatistics, [828](#)
- Task6
 - Ipopt::TimingStatistics, [826](#)
- Task6_
 - Ipopt::TimingStatistics, [828](#)
- tau_initialized_
 - Ipopt::IpoptData, [387](#)
- tau_min_
 - Ipopt::AdaptiveMuUpdate, [60](#)
 - Ipopt::MonotoneMuUpdate, [552](#)
- TauInitialized
 - Ipopt::IpoptData, [382](#)
- tcc_psi_
 - Ipopt::InexactPDSolver, [316](#)
 - Ipopt::InexactPDTerminationTester, [320](#)
- tcc_theta_
 - Ipopt::InexactLSAcceptor, [303](#)
 - Ipopt::InexactPDSolver, [316](#)
 - Ipopt::InexactPDTerminationTester, [320](#)
- tcc_theta_mu_exponent_
 - Ipopt::InexactPDSolver, [316](#)
 - Ipopt::InexactPDTerminationTester, [321](#)
- tcc_zeta_
 - Ipopt::InexactPDTerminationTester, [321](#)
- term_spaces_
 - Ipopt::SumMatrixSpace, [787](#)
 - Ipopt::SumSymMatrixSpace, [792](#)
- test_status_
 - Ipopt::CGPerturbationHandler, [145](#)
 - Ipopt::PDPerturbationHandler, [641](#)

- TestOrigProgress
 - Ipopt::RestoConvergenceCheck, [695](#)
 - Ipopt::RestoFilterConvergenceCheck, [699](#)
 - Ipopt::RestoPenaltyConvergenceCheck, [720](#)
- TestTermination
 - Ipopt::InexactNormalTerminationTester, [312](#)
 - Ipopt::InexactPDTerminationTester, [320](#)
 - Ipopt::IterativeSolverTerminationTester, [429](#)
- theta_max_
 - Ipopt::FilterLSAceptor, [253](#)
- theta_max_fact_
 - Ipopt::FilterLSAceptor, [253](#)
- theta_min_
 - Ipopt::CGPenaltyLSAceptor, [136](#)
 - Ipopt::FilterLSAceptor, [253](#)
- theta_min_fact_
 - Ipopt::FilterLSAceptor, [253](#)
- thresh
 - ma77_control_d, [502](#)
- TimedTask
 - Ipopt::TimedTask, [820](#)
- timing_statistics_
 - Ipopt::IpoptData, [388](#)
- TimingStatistics
 - Ipopt::TimingStatistics, [824](#)
- TimingStats
 - Ipopt::IpoptData, [385](#)
- tiny_step_flag
 - Ipopt::IpoptData, [383](#)
- tiny_step_flag_
 - Ipopt::IpoptData, [387](#)
- tiny_step_last_iteration_
 - Ipopt::BacktrackingLineSearch, [106](#)
- tiny_step_tol_
 - Ipopt::BacktrackingLineSearch, [104](#)
- tiny_step_y_tol_
 - Ipopt::BacktrackingLineSearch, [104](#)
- Tmp_c
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- tmp_c_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- Tmp_d
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- tmp_d_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- Tmp_s
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- tmp_s_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- Tmp_s_L_
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- tmp_s_L_
 - Ipopt::IpoptCalculatedQuantities, [374](#)
- Tmp_s_U_
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- tmp_s_U_
 - Ipopt::IpoptCalculatedQuantities, [374](#)
- tmp_slack_s_L_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_slack_s_U_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_slack_x_L_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_slack_x_U_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_step_s_L_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_step_s_U_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_step_v_L_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_step_v_U_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_step_x_L_
 - Ipopt::QualityFunctionMuOracle, [670](#)
- tmp_step_x_U_
 - Ipopt::QualityFunctionMuOracle, [670](#)
- tmp_step_z_L_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_step_z_U_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_v_L_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_v_U_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- Tmp_x
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- tmp_x_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- Tmp_x_L_
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- tmp_x_L_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- Tmp_x_U_
 - Ipopt::IpoptCalculatedQuantities, [364](#)
- tmp_x_U_
 - Ipopt::IpoptCalculatedQuantities, [374](#)
- tmp_z_L_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tmp_z_U_
 - Ipopt::QualityFunctionMuOracle, [671](#)
- tnlp
 - Ipopt::TNLPAdapter, [844](#)
- tnlp_
 - Ipopt::TNLPAdapter, [845](#)
 - Ipopt::TNLPReducer, [857](#)
- tol
 - Ipopt::IpoptData, [383](#)

- tol_
 - Ipopt::IpoptData, [387](#)
 - Ipopt::TNLPAdapter, [847](#)
- total_cpu_time_
 - Ipopt::SolveStatistics, [745](#)
- total_cputime_
 - Ipopt::TimedTask, [821](#)
- total_sys_time_
 - Ipopt::SolveStatistics, [745](#)
- total_systime_
 - Ipopt::TimedTask, [822](#)
- total_wallclock_time_
 - Ipopt::SolveStatistics, [745](#)
- total_walltime_
 - Ipopt::TimedTask, [822](#)
- TotalCPUTime
 - Ipopt::SolveStatistics, [744](#)
- TotalCpuTime
 - Ipopt::SolveStatistics, [744](#)
 - Ipopt::TimedTask, [821](#)
- TotalFunctionEvaluationCpuTime
 - Ipopt::OrigIpoptNLP, [615](#)
- TotalFunctionEvaluationSysTime
 - Ipopt::OrigIpoptNLP, [615](#)
- TotalFunctionEvaluationWallclockTime
 - Ipopt::OrigIpoptNLP, [615](#)
- TotalSysTime
 - Ipopt::SolveStatistics, [744](#)
 - Ipopt::TimedTask, [821](#)
- TotalWallclockTime
 - Ipopt::SolveStatistics, [744](#)
 - Ipopt::TimedTask, [821](#)
- TransMultVector
 - Ipopt::Matrix, [536](#)
- TransMultVectorImpl
 - Ipopt::CompoundMatrix, [154](#)
 - Ipopt::DenseGenMatrix, [192](#)
 - Ipopt::ExpandedMultiVectorMatrix, [230](#)
 - Ipopt::ExpansionMatrix, [236](#)
 - Ipopt::GenTMatrix, [267](#)
 - Ipopt::Matrix, [537](#)
 - Ipopt::MultiVectorMatrix, [557](#)
 - Ipopt::ScaledMatrix, [727](#)
 - Ipopt::SumMatrix, [783](#)
 - Ipopt::SymMatrix, [797](#)
 - Ipopt::TransposeMatrix, [861](#)
 - Ipopt::ZeroMatrix, [914](#)
 - Ipopt::ZeroSymMatrix, [918](#)
- TransposeMatrix
 - Ipopt::TransposeMatrix, [860](#)
- TransposeMatrixSpace
 - Ipopt::TransposeMatrixSpace, [863](#)
- tree_nodes
 - ma77_info_d, [506](#)
- trial
 - Ipopt::IpoptData, [380](#)
- trial_
 - Ipopt::IpoptData, [386](#)
- trial_avg_compl
 - Ipopt::IpoptCalculatedQuantities, [363](#)
- trial_avg_compl_cache_
 - Ipopt::IpoptCalculatedQuantities, [373](#)
- trial_barrier_obj
 - Ipopt::IpoptCalculatedQuantities, [357](#)
- trial_barrier_obj_cache_
 - Ipopt::IpoptCalculatedQuantities, [367](#)
- trial_c
 - Ipopt::IpoptCalculatedQuantities, [357](#)
- trial_c_cache_
 - Ipopt::IpoptCalculatedQuantities, [368](#)
- trial_compl_s_L
 - Ipopt::IpoptCalculatedQuantities, [360](#)
- trial_compl_s_L_cache_
 - Ipopt::IpoptCalculatedQuantities, [371](#)
- trial_compl_s_U
 - Ipopt::IpoptCalculatedQuantities, [361](#)
- trial_compl_s_U_cache_
 - Ipopt::IpoptCalculatedQuantities, [371](#)
- trial_compl_x_L
 - Ipopt::IpoptCalculatedQuantities, [360](#)
- trial_compl_x_L_cache_
 - Ipopt::IpoptCalculatedQuantities, [370](#)
- trial_compl_x_U
 - Ipopt::IpoptCalculatedQuantities, [360](#)
- trial_compl_x_U_cache_
 - Ipopt::IpoptCalculatedQuantities, [371](#)
- trial_complementarity
 - Ipopt::IpoptCalculatedQuantities, [361](#)
- trial_complementarity_cache_
 - Ipopt::IpoptCalculatedQuantities, [372](#)
- trial_constraint_violation
 - Ipopt::IpoptCalculatedQuantities, [359](#)
- trial_constraint_violation_cache_
 - Ipopt::IpoptCalculatedQuantities, [369](#)
- trial_d
 - Ipopt::IpoptCalculatedQuantities, [358](#)
- trial_d_cache_
 - Ipopt::IpoptCalculatedQuantities, [368](#)
- trial_d_minus_s
 - Ipopt::IpoptCalculatedQuantities, [358](#)
- trial_d_minus_s_cache_
 - Ipopt::IpoptCalculatedQuantities, [368](#)
- trial_dual_infeasibility
 - Ipopt::IpoptCalculatedQuantities, [361](#)
- trial_dual_infeasibility_cache_
 - Ipopt::IpoptCalculatedQuantities, [371](#)
- trial_f
 - Ipopt::IpoptCalculatedQuantities, [357](#)

- trial_f_cache_
 - Ipopt::IpoptCalculatedQuantities, 367
- trial_grad_f
 - Ipopt::IpoptCalculatedQuantities, 357
- trial_grad_f_cache_
 - Ipopt::IpoptCalculatedQuantities, 367
- trial_grad_lag_s
 - Ipopt::IpoptCalculatedQuantities, 360
- trial_grad_lag_s_cache_
 - Ipopt::IpoptCalculatedQuantities, 370
- trial_grad_lag_x
 - Ipopt::IpoptCalculatedQuantities, 360
- trial_grad_lag_x_cache_
 - Ipopt::IpoptCalculatedQuantities, 370
- trial_jac_c
 - Ipopt::IpoptCalculatedQuantities, 358
- trial_jac_c_cache_
 - Ipopt::IpoptCalculatedQuantities, 368
- trial_jac_cT_times_trial_y_c
 - Ipopt::IpoptCalculatedQuantities, 359
- trial_jac_cT_times_vec
 - Ipopt::IpoptCalculatedQuantities, 358
- trial_jac_cT_times_vec_cache_
 - Ipopt::IpoptCalculatedQuantities, 369
- trial_jac_d
 - Ipopt::IpoptCalculatedQuantities, 358
- trial_jac_d_cache_
 - Ipopt::IpoptCalculatedQuantities, 368
- trial_jac_dT_times_trial_y_d
 - Ipopt::IpoptCalculatedQuantities, 359
- trial_jac_dT_times_vec
 - Ipopt::IpoptCalculatedQuantities, 358
- trial_jac_dT_times_vec_cache_
 - Ipopt::IpoptCalculatedQuantities, 369
- trial_penalty_function
 - Ipopt::CGPenaltyCq, 120
- trial_penalty_function_cache_
 - Ipopt::CGPenaltyCq, 121
- trial_primal_dual_system_error
 - Ipopt::IpoptCalculatedQuantities, 362
- trial_primal_dual_system_error_cache_
 - Ipopt::IpoptCalculatedQuantities, 372
- trial_primal_infeasibility
 - Ipopt::IpoptCalculatedQuantities, 361
- trial_primal_infeasibility_cache_
 - Ipopt::IpoptCalculatedQuantities, 371
- trial_slack_s_L
 - Ipopt::IpoptCalculatedQuantities, 356
- trial_slack_s_L_cache_
 - Ipopt::IpoptCalculatedQuantities, 367
- trial_slack_s_U
 - Ipopt::IpoptCalculatedQuantities, 356
- trial_slack_s_U_cache_
 - Ipopt::IpoptCalculatedQuantities, 367
- trial_slack_x_L
 - Ipopt::IpoptCalculatedQuantities, 356
- trial_slack_x_L_cache_
 - Ipopt::IpoptCalculatedQuantities, 366
- trial_slack_x_U
 - Ipopt::IpoptCalculatedQuantities, 356
- trial_slack_x_U_cache_
 - Ipopt::IpoptCalculatedQuantities, 366
- TrialStatus
 - Ipopt::CGPerturbationHandler, 142
 - Ipopt::PDPerturbationHandler, 638
- Triangular_Format
 - Ipopt::TripletToCSRConverter, 872
- Triplet_Format
 - Ipopt::SparseSymLinearSolverInterface, 750
- triplet_to_csr_converter_
 - Ipopt::TSymLinearSolver, 883
- TripletEntry
 - Ipopt::TripletToCSRConverter::TripletEntry, 865
- TripletToCSRConverter
 - Ipopt::TripletToCSRConverter, 872
- try_tt2_
 - Ipopt::InexactPDTerminationTester, 323
- TryCorrector
 - Ipopt::BacktrackingLineSearch, 102
 - Ipopt::BacktrackingLSAceptor, 110
 - Ipopt::CGPenaltyLSAceptor, 133
 - Ipopt::FilterLSAceptor, 252
 - Ipopt::InexactLSAceptor, 301
 - Ipopt::PenaltyLSAceptor, 652
 - Ipopt::TimingStatistics, 826
- TryCorrector_
 - Ipopt::TimingStatistics, 828
- TrySecondOrderCorrection
 - Ipopt::BacktrackingLineSearch, 102
 - Ipopt::BacktrackingLSAceptor, 110
 - Ipopt::CGPenaltyLSAceptor, 133
 - Ipopt::FilterLSAceptor, 251
 - Ipopt::InexactLSAceptor, 301
 - Ipopt::PenaltyLSAceptor, 652
- TrySoftRestoStep
 - Ipopt::BacktrackingLineSearch, 102
- tsym_linear_solver_
 - Ipopt::TSymDependencyDetector, 877
- tt_eps2_
 - Ipopt::InexactPDTerminationTester, 321
- tt_eps3_
 - Ipopt::InexactPDTerminationTester, 321
- tt_kappa1_
 - Ipopt::InexactPDTerminationTester, 321
- tt_kappa2_
 - Ipopt::InexactPDTerminationTester, 321
- Type
 - Ipopt::AmplOptionsList::AmplOption, 71

- Ipopt::RegisteredOption, [682](#)
- type_
 - Ipopt::AmplOptionsList::AmplOption, [71](#)
 - Ipopt::IpoptException, [391](#)
 - Ipopt::RegisteredOption, [686](#)
- U
- u
 - ma77_control_d, [503](#)
 - ma86_control_d, [513](#)
 - ma97_control_d, [522](#)
- UNASSIGNED
 - Ipopt, [48](#)
- USER_REQUESTED_STOP
 - Ipopt, [48](#)
- USER_STOP
 - Ipopt::ConvergenceCheck, [181](#)
- U_
 - Ipopt::LimMemQuasiNewtonUpdater, [456](#)
 - Ipopt::LowRankUpdateSymMatrix, [481](#)
- U_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [458](#)
- umax_
 - Ipopt::Ma77SolverInterface, [512](#)
 - Ipopt::Ma86SolverInterface, [520](#)
 - Ipopt::Ma97SolverInterface, [531](#)
- umin
 - ma77_control_d, [503](#)
 - ma86_control_d, [513](#)
- unapply_grad_obj_scaling
 - Ipopt::NLPScalingObject, [587](#)
- unapply_grad_obj_scaling_NonConst
 - Ipopt::NLPScalingObject, [587](#)
- unapply_obj_scaling
 - Ipopt::NLPScalingObject, [584](#)
 - Ipopt::StandardScalingBase, [756](#)
- unapply_vector_scaling_c
 - Ipopt::NLPScalingObject, [585](#)
 - Ipopt::StandardScalingBase, [756](#)
- unapply_vector_scaling_c_NonConst
 - Ipopt::NLPScalingObject, [585](#)
 - Ipopt::StandardScalingBase, [757](#)
- unapply_vector_scaling_d
 - Ipopt::NLPScalingObject, [585](#)
 - Ipopt::StandardScalingBase, [757](#)
- unapply_vector_scaling_d_LU
 - Ipopt::NLPScalingObject, [586](#)
- unapply_vector_scaling_d_LU_NonConst
 - Ipopt::NLPScalingObject, [586](#)
- unapply_vector_scaling_d_NonConst
 - Ipopt::NLPScalingObject, [585](#)
 - Ipopt::StandardScalingBase, [757](#)
- unapply_vector_scaling_x
 - Ipopt::NLPScalingObject, [584](#)
- Ipopt::StandardScalingBase, [756](#)
- unapply_vector_scaling_x_NonConst
 - Ipopt::NLPScalingObject, [584](#)
- Ipopt::StandardScalingBase, [756](#)
- uncached_dual_frac_to_the_bound
 - Ipopt::IpoptCalculatedQuantities, [362](#)
- uncached_slack_frac_to_the_bound
 - Ipopt::IpoptCalculatedQuantities, [363](#)
- uninitialized_h
 - Ipopt::IpoptNLP, [398](#)
 - Ipopt::OrigIpoptNLP, [612](#)
 - Ipopt::RestIpoptNLP, [706](#)
- unit_diagnostics
 - ma77_control_d, [501](#)
 - ma86_control_d, [513](#)
 - ma97_control_d, [522](#)
- unit_error
 - ma77_control_d, [501](#)
 - ma86_control_d, [513](#)
 - ma97_control_d, [522](#)
- unit_restart
 - ma77_info_d, [506](#)
- unit_warning
 - ma77_control_d, [501](#)
 - ma86_control_d, [513](#)
 - ma97_control_d, [522](#)
- Unrecoverable_Exception
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [49](#)
- UnscaleSigma
 - Ipopt::QualityFunctionMuOracle, [669](#)
- unscaled_curr_c
 - Ipopt::IpoptCalculatedQuantities, [357](#)
- unscaled_curr_complementarity
 - Ipopt::IpoptCalculatedQuantities, [361](#)
- unscaled_curr_d
 - Ipopt::IpoptCalculatedQuantities, [358](#)
- unscaled_curr_dual_infeasibility
 - Ipopt::IpoptCalculatedQuantities, [361](#)
- unscaled_curr_dual_infeasibility_cache_
 - Ipopt::IpoptCalculatedQuantities, [371](#)
- unscaled_curr_f
 - Ipopt::IpoptCalculatedQuantities, [356](#)
- unscaled_curr_nlp_constraint_violation
 - Ipopt::IpoptCalculatedQuantities, [359](#)
- unscaled_curr_nlp_constraint_violation_cache_
 - Ipopt::IpoptCalculatedQuantities, [369](#)
- unscaled_curr_nlp_error
 - Ipopt::IpoptCalculatedQuantities, [362](#)
- unscaled_curr_nlp_error_cache_
 - Ipopt::IpoptCalculatedQuantities, [372](#)
- unscaled_matrix_space_
 - Ipopt::ScaledMatrixSpace, [731](#)
 - Ipopt::SymScaledMatrixSpace, [806](#)

- unscaled_trial_c
 - Ipopt::IpoptCalculatedQuantities, [357](#)
- unscaled_trial_f
 - Ipopt::IpoptCalculatedQuantities, [357](#)
- unscaled_trial_nlp_constraint_violation
 - Ipopt::IpoptCalculatedQuantities, [359](#)
- unscaled_trial_nlp_constraint_violation_cache_
 - Ipopt::IpoptCalculatedQuantities, [369](#)
- unscaled_x_cache_
 - Ipopt::OrigIpoptNLP, [618](#)
- UnscaledMatrixSpace
 - Ipopt::ScaledMatrixSpace, [731](#)
 - Ipopt::SymScaledMatrixSpace, [806](#)
- unused
 - ma77_info_d, [506](#)
- update_for_resto_
 - Ipopt::LimMemQuasiNewtonUpdater, [455](#)
- update_local_lambda
 - Ipopt::TNLPAdapter, [845](#)
- update_local_x
 - Ipopt::TNLPAdapter, [845](#)
- UpdateBarrierParameter
 - Ipopt::AdaptiveMuUpdate, [59](#)
 - Ipopt::IpoptAlgorithm, [336](#)
 - Ipopt::MonotoneMuUpdate, [552](#)
 - Ipopt::MuUpdate, [570](#)
 - Ipopt::TimingStatistics, [825](#)
- UpdateBarrierParameter_
 - Ipopt::TimingStatistics, [827](#)
- UpdateEntry
 - Ipopt::PiecewisePenalty, [657](#)
- UpdateExtendedData
 - Ipopt::LowRankSSAugSystemSolver, [473](#)
- UpdateFactorization
 - Ipopt::LowRankAugSystemSolver, [467](#)
- UpdateForNextIteration
 - Ipopt::BacktrackingLSAcceptor, [110](#)
 - Ipopt::CGPenaltyLSAcceptor, [133](#)
 - Ipopt::FilterLSAcceptor, [252](#)
 - Ipopt::InexactLSAcceptor, [301](#)
 - Ipopt::PenaltyLSAcceptor, [652](#)
- UpdateHessian
 - Ipopt::ExactHessianUpdater, [227](#)
 - Ipopt::HessianUpdater, [276](#)
 - Ipopt::IpoptAlgorithm, [336](#)
 - Ipopt::LimMemQuasiNewtonUpdater, [451](#)
 - Ipopt::TimingStatistics, [825](#)
- UpdateHessian_
 - Ipopt::TimingStatistics, [827](#)
- UpdateInternalData
 - Ipopt::LimMemQuasiNewtonUpdater, [452](#)
- UpdatePenaltyParameter
 - Ipopt::CGPenaltyLSAcceptor, [135](#)
- UpdateTags
 - Ipopt::GenAugSystemSolver, [259](#)
- upper_
 - Ipopt::RegisteredOption, [686](#)
- upper_strict_
 - Ipopt::RegisteredOption, [686](#)
- UpperInteger
 - Ipopt::RegisteredOption, [683](#)
- UpperNumber
 - Ipopt::RegisteredOption, [683](#)
- UpperStrict
 - Ipopt::RegisteredOption, [683](#)
- use_scaling_
 - Ipopt::TSymLinearSolver, [882](#)
- User_Requested_Stop
 - IpReturnCodes_inc.h, [998](#)
 - Ipopt, [48](#)
- user_data
 - IpStdCInterface.h, [1006](#)
- user_data_
 - Ipopt::StdInterfaceTNLP, [774](#)
- UserDataPtr
 - IpStdCInterface.h, [1002](#)
- UserScaling
 - Ipopt::UserScaling, [886](#)
- usmall
 - ma77_info_d, [506](#)
 - ma86_info_d, [515](#)
- Utilde2_
 - Ipopt::LowRankAugSystemSolver, [469](#)
- V
- V_
 - Ipopt::LimMemQuasiNewtonUpdater, [456](#)
 - Ipopt::LowRankUpdateSymMatrix, [481](#)
- v_L
 - Ipopt::IteratesVector, [409](#)
- v_L_NonConst
 - Ipopt::IteratesVector, [409](#)
- v_L_space_
 - Ipopt::IteratesVectorSpace, [415](#)
- v_U
 - Ipopt::IteratesVector, [409](#)
- v_U_NonConst
 - Ipopt::IteratesVector, [410](#)
- v_U_space_
 - Ipopt::IteratesVectorSpace, [415](#)
- v_norm_scaled_
 - Ipopt::InexactPDTerminationTester, [322](#)
- V_old_
 - Ipopt::LimMemQuasiNewtonUpdater, [458](#)
- VPrintf
 - Ipopt::Journalist, [442](#)
- VPrintfIndented
 - Ipopt::Journalist, [442](#)

- val
 - Ipopt::FilterEntry, [246](#)
 - IpStdCInterface.h, [1005](#)
- val_
 - Ipopt::Ma77SolverInterface, [511](#)
 - Ipopt::Ma86SolverInterface, [520](#)
 - Ipopt::Ma97SolverInterface, [531](#)
- valid_cache_tag_
 - Ipopt::Matrix, [539](#)
 - Ipopt::Vector, [899](#)
- valid_strings_
 - Ipopt::RegisteredOption, [686](#)
- vals_
 - Ipopt::FilterEntry, [246](#)
- Value
 - Ipopt::OptionsList::OptionValue, [604](#)
- value_
 - Ipopt::OptionsList::OptionValue, [604](#)
 - Ipopt::RegisteredOption::string_entry, [778](#)
- Values
 - Ipopt::DenseGenMatrix, [191](#)
 - Ipopt::DenseSymMatrix, [197](#)
 - Ipopt::DenseVector, [204](#)
 - Ipopt::GenTMatrix, [267](#)
 - Ipopt::SymTMatrix, [809](#)
- values_
 - Ipopt::DenseGenMatrix, [193](#)
 - Ipopt::DenseSymMatrix, [199](#)
 - Ipopt::DenseVector, [208](#)
 - Ipopt::GenTMatrix, [269](#)
 - Ipopt::SymTMatrix, [810](#)
- values_allocated
 - Ipopt::DenseVector, [208](#)
- var_integer_md_
 - Ipopt::AmplTNLP, [87](#)
- var_numeric_md_
 - Ipopt::AmplTNLP, [87](#)
- var_string_md_
 - Ipopt::AmplTNLP, [87](#)
- Variable_Source
 - Ipopt::AmplSuffixHandler, [76](#)
- vartheta_
 - Ipopt::CGSearchDirCalculator, [150](#)
- Vec
 - Ipopt::MultiVectorMatrix, [558](#)
- vec_space_
 - Ipopt::ExpandedMultiVectorMatrixSpace, [233](#)
 - Ipopt::MultiVectorMatrixSpace, [560](#)
- vecs_
 - Ipopt::ExpandedMultiVectorMatrix, [231](#)
- Vector
 - Ipopt::Vector, [891](#)
- VectorSpace
 - Ipopt::VectorSpace, [900](#)
- vectors_valid_
 - Ipopt::CompoundVector, [176](#)
- VectorsValid
 - Ipopt::CompoundVector, [175](#)
- Vtilde1_
 - Ipopt::LowRankAugSystemSolver, [469](#)
- W
 - Ipopt::IpoptData, [381](#)
- WS_Option
 - Ipopt::AmplOptionsList, [73](#)
- W_
 - Ipopt::IpoptData, [386](#)
- w_factor_
 - Ipopt::GenAugSystemSolver, [260](#)
 - Ipopt::LowRankAugSystemSolver, [468](#)
 - Ipopt::LowRankSSAugSystemSolver, [474](#)
 - Ipopt::StdAugSystemSolver, [764](#)
- w_tag_
 - Ipopt::GenAugSystemSolver, [260](#)
 - Ipopt::LowRankAugSystemSolver, [467](#)
 - Ipopt::LowRankSSAugSystemSolver, [474](#)
 - Ipopt::StdAugSystemSolver, [764](#)
- WallclockTime
 - Ipopt, [52](#)
- warm_start_bound_frac_
 - Ipopt::WarmStartIterateInitializer, [904](#)
- warm_start_bound_push_
 - Ipopt::WarmStartIterateInitializer, [904](#)
- warm_start_entire_iterate_
 - Ipopt::WarmStartIterateInitializer, [904](#)
- warm_start_init_point_
 - Ipopt::DefaultIterateInitializer, [187](#)
- warm_start_initializer_
 - Ipopt::DefaultIterateInitializer, [187](#)
- warm_start_mult_bound_push_
 - Ipopt::WarmStartIterateInitializer, [904](#)
- warm_start_mult_init_max_
 - Ipopt::WarmStartIterateInitializer, [904](#)
- warm_start_same_structure_
 - Ipopt::GenAugSystemSolver, [261](#)
 - Ipopt::IpoptCalculatedQuantities, [366](#)
 - Ipopt::Ma27TSolverInterface, [490](#)
 - Ipopt::Ma57TSolverInterface, [499](#)
 - Ipopt::MumpsSolverInterface, [566](#)
 - Ipopt::OrigIpoptNLP, [619](#)
 - Ipopt::StdAugSystemSolver, [766](#)
 - Ipopt::TNLPAdapter, [846](#)
 - Ipopt::TSymLinearSolver, [883](#)
- warm_start_slack_bound_frac_
 - Ipopt::WarmStartIterateInitializer, [904](#)
- warm_start_slack_bound_push_
 - Ipopt::WarmStartIterateInitializer, [904](#)

- warm_start_target_mu_
 - Ipopt::WarmStartIterateInitializer, 904
- WarmStartIterateInitializer
 - Ipopt::WarmStartIterateInitializer, 903
- watchdog_alpha_primal_test_
 - Ipopt::BacktrackingLineSearch, 105
- watchdog_barr_
 - Ipopt::FilterLSAcceptor, 255
 - Ipopt::InexactLSAcceptor, 304
 - Ipopt::PenaltyLSAcceptor, 654
- watchdog_delta_
 - Ipopt::BacktrackingLineSearch, 105
- watchdog_delta_cgpen_
 - Ipopt::CGPenaltyLSAcceptor, 138
- watchdog_direct_deriv_penalty_function_
 - Ipopt::CGPenaltyLSAcceptor, 138
- watchdog_gradBarrTDelta_
 - Ipopt::FilterLSAcceptor, 255
- watchdog_iterate_
 - Ipopt::BacktrackingLineSearch, 105
- watchdog_penalty_function_
 - Ipopt::CGPenaltyLSAcceptor, 138
- watchdog_pred_
 - Ipopt::InexactLSAcceptor, 304
 - Ipopt::PenaltyLSAcceptor, 654
- watchdog_shortened_iter_
 - Ipopt::BacktrackingLineSearch, 105
- watchdog_shortened_iter_trigger_
 - Ipopt::BacktrackingLineSearch, 105
- watchdog_theta_
 - Ipopt::FilterLSAcceptor, 255
 - Ipopt::InexactLSAcceptor, 304
 - Ipopt::PenaltyLSAcceptor, 654
- watchdog_trial_iter_
 - Ipopt::BacktrackingLineSearch, 105
- watchdog_trial_iter_max_
 - Ipopt::BacktrackingLineSearch, 105
- wd_cntl_
 - Ipopt::Ma57TSolverInterface, 499
- wd_fact_
 - Ipopt::Ma57TSolverInterface, 500
- wd_icntl_
 - Ipopt::Ma57TSolverInterface, 499
- wd_ifact_
 - Ipopt::Ma57TSolverInterface, 500
- wd_info_
 - Ipopt::Ma57TSolverInterface, 499
- wd_iwork_
 - Ipopt::Ma57TSolverInterface, 500
- wd_keep_
 - Ipopt::Ma57TSolverInterface, 500
- wd_lfact_
 - Ipopt::Ma57TSolverInterface, 500
- wd_lifact_
 - Ipopt::Ma57TSolverInterface, 500
- wd_lkeep_
 - Ipopt::Ma57TSolverInterface, 500
- wd_rinfo_
 - Ipopt::Ma57TSolverInterface, 499
- Wdiag_
 - Ipopt::LowRankAugSystemSolver, 469
 - Ipopt::LowRankSSAugSystemSolver, 475
- will_allow_clobber
 - Ipopt::OptionsList, 601
- wp
 - mc68_control, 544
- write_solution_file
 - Ipopt::AmplTNLP, 84
- WriteOutput
 - Ipopt::IterationOutput, 417
 - Ipopt::OrigIterationOutput, 622
 - Ipopt::RestIterationOutput, 717
- wsmp_inexact_droptol_
 - Ipopt::IterativeWsmpSolverInterface, 435
- wsmp_inexact_fillin_limit_
 - Ipopt::IterativeWsmpSolverInterface, 435
- wsmp_no_pivoting_
 - Ipopt::WsmpSolverInterface, 910
- wsmp_num_threads_
 - Ipopt::IterativeWsmpSolverInterface, 435
 - Ipopt::WsmpSolverInterface, 910
- wsmp_pivotol_
 - Ipopt::IterativeWsmpSolverInterface, 435
 - Ipopt::WsmpSolverInterface, 910
- wsmp_pivotolmax_
 - Ipopt::IterativeWsmpSolverInterface, 435
 - Ipopt::WsmpSolverInterface, 910
- wsmp_scaling_
 - Ipopt::IterativeWsmpSolverInterface, 435
 - Ipopt::WsmpSolverInterface, 910
- wsmp_singularity_threshold_
 - Ipopt::WsmpSolverInterface, 910
- wsmp_write_matrix_iteration_
 - Ipopt::IterativeWsmpSolverInterface, 435
 - Ipopt::WsmpSolverInterface, 910
- WsmpSolverInterface
 - Ipopt::WsmpSolverInterface, 908
- X
- x
 - Ipopt::IteratesVector, 405
 - IpStdCInterface.h, 1006
- x_L
 - Ipopt::IpoptNLP, 395
 - Ipopt::OrigIpoptNLP, 612
 - Ipopt::RestIpoptNLP, 706
 - IpStdCInterface.h, 1004
- x_L_

- Ipopt::OrigIpoptNLP, 618
 - Ipopt::RestIpoptNLP, 710
 - Ipopt::StdInterfaceTNLP, 772
- x_NonConst
 - Ipopt::IteratesVector, 405
- x_U
 - Ipopt::IpoptNLP, 396
 - Ipopt::OrigIpoptNLP, 612
 - Ipopt::RestIpoptNLP, 706
 - IpStdCInterface.h, 1004
- x_U_
 - Ipopt::OrigIpoptNLP, 618
 - Ipopt::RestIpoptNLP, 710
 - Ipopt::StdInterfaceTNLP, 772
- x_fixed_map_
 - Ipopt::TNLPAdapter, 851
- x_l_space_
 - Ipopt::OrigIpoptNLP, 616
 - Ipopt::RestIpoptNLP, 709
 - Ipopt::TNLPAdapter, 848
- x_ref_
 - Ipopt::RestIpoptNLP, 711
- x_scaling
 - IpStdCInterface.h, 1006
- x_scaling_
 - Ipopt::StdInterfaceTNLP, 774
- x_sol_
 - Ipopt::AmplTNLP, 85
 - Ipopt::StdInterfaceTNLP, 774
- x_space_
 - Ipopt::IteratesVectorSpace, 414
 - Ipopt::OrigIpoptNLP, 616
 - Ipopt::RestIpoptNLP, 709
 - Ipopt::TNLPAdapter, 848
- x_tag_for_g_
 - Ipopt::TNLPAdapter, 850
- x_tag_for_iterates_
 - Ipopt::TNLPAdapter, 849
- x_tag_for_jac_g_
 - Ipopt::TNLPAdapter, 850
- x_u_space_
 - Ipopt::OrigIpoptNLP, 616
 - Ipopt::RestIpoptNLP, 709
 - Ipopt::TNLPAdapter, 848
- Y
- Y_
 - Ipopt::LimMemQuasiNewtonUpdater, 456
- y_c
 - Ipopt::IteratesVector, 406
- y_c_NonConst
 - Ipopt::IteratesVector, 406
- y_c_ext_space_
 - Ipopt::LowRankSSAugSystemSolver, 476
- y_c_space_
 - Ipopt::IteratesVectorSpace, 414
- y_c_tag_for_iterates_
 - Ipopt::TNLPAdapter, 849
- y_d
 - Ipopt::IteratesVector, 407
- y_d_NonConst
 - Ipopt::IteratesVector, 407
- y_d_space_
 - Ipopt::IteratesVectorSpace, 415
- y_d_tag_for_iterates_
 - Ipopt::TNLPAdapter, 850
- Y_old_
 - Ipopt::LimMemQuasiNewtonUpdater, 458
- Ypart_
 - Ipopt::LimMemQuasiNewtonUpdater, 456
- Ypart_old_
 - Ipopt::LimMemQuasiNewtonUpdater, 458
- Z
- z_L
 - Ipopt::IteratesVector, 407
- z_L_NonConst
 - Ipopt::IteratesVector, 408
- z_L_sol_
 - Ipopt::AmplTNLP, 86
 - Ipopt::StdInterfaceTNLP, 774
- z_L_space_
 - Ipopt::IteratesVectorSpace, 415
- z_U
 - Ipopt::IteratesVector, 408
- z_U_NonConst
 - Ipopt::IteratesVector, 408
- z_U_sol_
 - Ipopt::AmplTNLP, 86
 - Ipopt::StdInterfaceTNLP, 775
- z_U_space_
 - Ipopt::IteratesVectorSpace, 415
- zb01_info
 - mc68_info, 546
- ZeroMatrix
 - Ipopt::ZeroMatrix, 913
- ZeroMatrixSpace
 - Ipopt::ZeroMatrixSpace, 915, 916
- ZeroSymMatrix
 - Ipopt::ZeroSymMatrix, 918
- ZeroSymMatrixSpace
 - Ipopt::ZeroSymMatrixSpace, 920