







### 1.0.3 Inequality Constraints

NLPAddInequalityConstraint

## 1.2 Vectors

NLCreateVector	95
NLCreateVectorWithSparseData	96
NLCreateVectorWithFullData	98
NLCreateDenseWrappedVector	100
NLFreeVector	101
NLRefVector	102
NLPri4tVector	103
NLVGetNC	104
NLVGetC	105
NLVSetC	106
NLCopyVector	107
NLVSetToZero	108
NLVIncrementC	109
NLVInnerProd	110
NLVPlusV	111
NLNegateVector	112
NLVSparse	113
NLVnNonZeros	114
NLVnonZero	115
NLVGetNumberOfNonZeros	116
NLVGetNonZeroCoord	117
NLVGetNonZero	118roCoord

## 1.3 Matrices

NLCreateMatrix	121
NLCreateMatrixWithData	122
NLCreateSparseMatrix	124
NLCreateWSMPSparseMatrix	125
NLCreateDenseWrappedMatrix	127
NLRefMatrix	128
NLFreeMatrix	129
NLMGetNumberOfRows	130
NLMGetNumberOfColumns	131
NLMGetElement	132
NLMSetElement	133
NLMIncrementElement	134
NLMMatrixDoubleProduct	135
NLMVMult	136
NLMVMultT	137
NLMSetToZero	138
NLMMatrixClone	139
NLMGetGershgorinBounds	140
NLMMatrixOneNorm	141
NLMSumSubMatrixInto	142
NLMSumRankOneInto	143
NLMMPProd	144
NLMSparse	145
NLMDetermineHessianSparsityStructure	146
NLMData	147
NLMnE	148
NLMRow	149
NLMCol	150
NLPrintMatrix	151



### 1.5.2 The list of groups





#### 1.5.4 The list of Types

### 1.6.3 Setting LANCELOT Parameters

LNSetCheckDerivatives	229
LNGetCheckDerivatives	230
LNSetConcT/FeT/ai nT/FAccuracy	231
LNGetConcT/FeT/ai nT/FAccuracy	232
LNSetFircT/FCT/oncT/FeT/ai nT/FAccuracy	233
LNGetFircT/FCT/oncT/FeT/ai nT/FAccuracy	234

LNSetScalings	257
LNGetScalings	258
LNSetSolveBQPAccurately	259
LNGetSolveBQPAccurately	260
LNSetLinearSolverMethod	261
LNGetLinearSolverMethod	263
LNGetLinearSolverBandwidth	265
LNSetStopOnBadDerivatives	266
LNGetStopOnBadDerivatives	267
LNSetTrustRegionRd[(N)1S3(th)1(od)]T/F111.95Tf19a	

## NLCreateProblem

### Purpose

Allocates and initializes an NLProblem data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>  
P
```



## NLFreeProblem

### Purpose

Releases storage associated with an NLProblem data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
void NLFreeProblem(P);

NLProblem P The problem.
```

### Description

The routine





## NLPrintProblemShort

### Purpose

Prints a NLProblem data structure.

NLPGetProblemName

Purpose

## NLPGetNumberOfVariables

### Purpose

Returns the number of variables for a problem.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
n=NLPGetNumberOfVariables(P);

int      n   The number of variables.
NLProblem P  The problem.
```

### Description

This routine returns the number of variables for a problem. This is set when the `NLCreateProblem` (page 13) subroutine is called.

### Errors

Errors return -1.

Message	Severity
"Problem (argument 1) is NULL"	12

## NLPSetVariableScale

### Purpose

Sets the scale factor of a variable.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
rc
```



## NLPSetVariableName

---

### Purpose

Assigns the name of a variable.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
rc=NLPSetVariableName(P, i, name);
```

int	<i>rc</i>	The return code.
NLProblem	<i>P</i>	The problem.
int	<i>i</i>	The number of the variable.
char	<i>*name</i>	The problem name.

### Description

This routine sets the name of a variable. This may be queried with the NLPSetVariableName subroutine (page 22). If the variable has not yet been given a name, the default is "XXXXXXXX", where 'x' is a hex digit 0-9A-F. This is created with the C-format "X"

A copy of the string is made. The copy is freed when the problem is freed.

### Errors

Errors return 0 and make no changes to the problem. No2(o)15e

## NLPGetVariableName

### Purpose

Returns the name of a variable.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
name=NLPGetVariableName(P, i);
```

char	<i>*name</i>	The problem name.
NLProblem	<i>P</i>	The problem.
int	<i>i</i>	The number of the variable.

### Description

This routine returns the name of a variable. If the variable has not yet been given a name, the default is "XXXXXXXX", where 'x' is a hex digit 0-9A-F. This is create with the C-format "X"

Note: The user should not free the returned string.

### Errors

Errors return (char\*)NULL.

## NLPSetSimpleBounds

### Purpose

Sets the bounds on a variable.

### Library

libNLPAPI.a



## NLPSetLowerSimpleBound

## Purpose

Sets the lower bound on a variable.

## Library

libNLPAPI.a

## C Syntax

```
#include <NLPAPI.h>
```

```
rc=NLPSetLowerSimpleBound(P1(A),P2(A),P3(A))%F70767(P1imp)rimAuF7sets0dur(o)
```

## NLPGetLowerSimpleBound

### Purpose

Gets the lower bound on a variable.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
/=NLPGetLowerSimpleBound(
```

NLPISLowerSimpleBoundSet

Purpose

## NLPSetUpperSimpleBound

### Purpose

Sets the upper bound on a variable.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
rc=NLPSetUpperSimpleBound(P, var, u);

int      rc    The return code.
NLProblem P    The problem.
int      var   Which variable.
double   u    The upper bound.
```

### Description

This routine sets the upper bound on the variable. This can be queried with the NLPGetUpperSimpleBound (page 29) subroutine. The bounds can also be set at the same time (using the NLPSetSimpleBounds

the bound is . (A value of 1.e20 is considered to be infinity.)

turn 0 and make no changes to the problem. Normal execution

### Severity

"Problem (argument 1"1(u47)-2-3272(with)]TJ0-14.445"Vaeaia03eumiTd[(b)-28(e)11.9%7(ound)

## NLPGetUpperSimpleBound

### Purpose

Gets the upper bound on a variable.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
u=NLPGetUpperSimpleBound(P
```

## NLPIsUpperSimpleBoundSet

### Purpose

Queries whether a upper bound has been set on a variable.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
ans=NLPISUpperSimpleBoundSet(P, var);
```

int            *ans*    The answer is whether

int            *var*    The variable to be queried.

## NLConvertToEqualityAndBoundsOnly

### Purpose

Eliminates the inequality constraints from a Problem by introducing slack variables.

### Library

C Syntax

libNLPAPI.aLibrary

#include <NLPAPI.h>

NLConvertToEqualityAndBoundsOnly(*P*);

NLProblem





## NLCopyProblem

### Purpose

Creates a copy of an NLProblem data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
Q=NLCopyProblem(P);

    NLProblem P   The problem.
    NLProblem Q   The copy.
```

### Description

The routine NLCopyProblem makes a “shallow” copy of a problem. That is, the lists of constraints are duplicated, but the functions defining the objective and constraints (the group and element functions) are not.

### Errors

Severity 12 errors return (NLProblem)NULL, severity 4 returns a problem with no name.

Message	Severity
"Problem (argumes 1) is NULL"	4
"Out of memory, trying to allocate %d bytes"	12

NLCreateAugmentedLagrangian

Purpose

the objective. That is, a problem

minimize  $O(\mathbf{v})$

$$f_i(\mathbf{v}) = 0$$

## NLSetLambdaAndMuInAugmentedLagrangian

### Purpose

Sets the penalty parameter and LLagrange multipliers in a Problem with a quadratic penalty and Lagrangian terms in the objective.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
NLSetLambdaAndMuInAugmentedLagrangian(
```

Message

Severity

---

NSIM3000Fility constraints (nt and removes the bou8ds.  
by creating a Dility constraints







## NLPSetObjectiveByString

### Purpose

Sets the objective to be a function defined by a string.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
NLPSetObjectiveByString(P, name, nv, v, varlist, expr);
```

NLPProblem *P*    The problem.

char\* *name*    A name given to the objective ("Obj" might be a good choice.)

int *nv*    The dimension of the domain of the objective. This provides some degree of sparsity.

int\* *v*    A list of length *nv* of the problem variables that the objective depends on.

char\* *varlist*    A list of identifiers in the *expr*. When the expression is evaluated these identifiers will be given the values of the problem variables listed in *v* (in the same order). The list is a single string, delimited by the characters "[" and "]", and separated by commas.

char\* *expr*    An expression giving the objective.

### Description

will assign  $a$  the value of problem variable 1,  $c$  the value of problem variable 45, and



## NLPAddNonlinearElementToObjectiveGroup

### Purpose

NLPSetObjectiveGroupA

Purpose

## NLPSetObjectiveGroupB

### Purpose

Sets the constant part of the linear element of a group in the objective.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
rc=NLPSetObjectiveGroupB(
    NA7eu0ldm0
    in0
```









## NLPGetObjectiveGroupNumber

### Purpose

Returns the index of a group in the objective of a problem.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

NLPEvaluateObjective

Purpose

## C Syntax

```
#include <NLPAPI.h>
rc=NLPEvaluateGradientOfObjective( $P, x, g$ );
int rc
```

## NLPEvaluateHessianOfObjective

### Purpose

Evaluates the Hessian of the objective function.

Lib6.64.d2p.64.a111.95Tf021.646TdI(Li)1NLI

NLPAddEqualityConstraint

Purpose

The first argument to  $f$  will be  $nv$ . The second argument is an array  $x$

## NLPAddEqualityConstraintByString

### Purpose

Adds an equality constraint defined by an expression in a string to a problem.

### Library

`libNLPAPI.a`

### C Syntax

```
#include <NLPAPI.h>
```



```
c=NLPAddEqualityConstraintByString(P,"Obj",3,v, "[a,b,c]", "a**2+sqrt(cos(b))+1./
```

## NLPAddNonlinearEqualityConstraint

### Purpose

Adds a nonlinear equality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

*g*



## NLPAddNonlinearElementToEqualityConstraint

### Purpose

Adds an empty nonlinear element to an equality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
e=NLPAddNonlinearElementToEqualityConstraint(P, constraint, weight, ne, variables, xfrm);  
int NLPAddNonlinearElementToEqualityConstraint(NLPProblemNoi1(onstr)50(aint.1(o)1(nstr-
```

## NLPSetEqualityConstraintA

### Purpose

Sets the linear part of the linear element of an equality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
rc=NLPSetEqualityConstraintA(0, const0(c), 1, 1);

int      rc      The return code.
NLProblem P      The problem.
int      const0(c) The index of the constraint.
NLVector a      The linear element.
```

### Description

This routine sets the linear part of the linear element of an equality constraint.

## NLPSetEqualityConstraintB

### Purpose

Sets the constant part of the linear element of an equality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
rc=NLPSetEqualityConstraintB(P, constraint, b);
```

int	<i>rc</i>	The return code.
NLProblem	<i>P</i>	The problem.
int	<i>constraint</i>	The index of the constraint.
double	<i>b</i>	The constant.

### Description

This routine sets the constant part of the linear element of an equality constraint. The default value is zero.

Note:

### Errors

ErTJ/F2rs return 0 and make no changes to the problem. Normal execution returns 1.

Message	Severity
"Problem (argument 1) is NULL"	12
"Group %d is illegal (argument 2). Must be in range 0 to %d"	12



NLPGetEqualityConstraintGroupNumber

Purpose



## NLPEvaluateEqualityConstraint

### Purpose

Evaluates an equality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
0
```





## NLPAddInequalityConstraint

### Purpose

Adds an inequality constraint defined by a subroutine (and it's derivatives) to a problem.

### Library

C Syntax

libNLPAPI.aLibrary

#include <NLPAPI.h>

```
c=NLPAddInequalityConstraint(P, name, l, u, nv, v, f, df, ddf, data, freeData);
```

int *c*

NLProblem *P*

char\* *name*

double *l*

The number assigned to the new constraint.

The problem.

A name given to the constraint.

*c*



NLPAddInequalityConstraintByString

Purpose

```
int v[3];  
int c;
```

```
v[0]=1; v[1]=45; v[2]=0;
```

```
c=NLPAddInequalityConstraintByString(P, "Obj ", -2. e20, 1. , 3, v, "[a, b, c]", "a**2+sq
```

will assign a the value of problem variable 1, c the value of problem variable 45, and c the value of problem variable 0. The main restriction on the expression is that constants may *not* be specified using exponential notation (sorry). Elementary functions and the usual binary operations can be used. Automatic differentiation is used to evaluate the derivatives.

## Errors

Message	Severity
"Problem (first arg.) is NULL, you must supply a problem."	12
"name (second arg.) is NULL, you must supply a name for the constraint."	12







## NLPAddNonlinearElementToInequalityConstraint

### Purpose

Adds an empty nonlinear element to an inequality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
e=NLPAddNonlinearElementToInequalityConstraint(P, constraint, weight, ne, variables, xfrm);
```

```
int NLPAddNonlinearElementToInequalityConstraint(NLPProblem P,int con-  
straint,double w,LNNonlinearElement E);
```

int	<i>e</i>	The index of the new nonlinear element.
NLPProblem	<i>P</i>	The problem.
int	<i>constraint</i>	The index of the constraint.
double	<i>weight</i>	The weight.
LNNonlinearElement	<i>ne</i>	

## NLPSetInequalityConstraintA

### Purpose

Sets the linear part of the linear element of an inequality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
rc=NLPSetInequalityConstraintA(P, constraint, a);
```

int rc      The return code  
int *constraint*

NLPSetInequalityConstraintB

Purpose



## NLPSetInequalityConstraintLowerBound

### Purpose

Sets the lower bound on an inequality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
rc=NLPSetInequalityConstraintLowerBound( $P$ ,  $c$ ,  $l$ );
```

```
int rc
```

## NLPGetInequalityConstraintUpperBound

### Purpose

Gets the upper bound for an inequality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
u=NLPGetInequalityConstraintUpperBound(P, c);

double      u   The upper bound.
NLProblem   P   The problem.
int         c   Which constraint.
```

### Description

## NLPSetInequalityConstraintUpperBound

### Purpose

Sets the upper bound on an inequality constraint.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
rc=NLPSetInequalityConstraintUpperBound(P, c, u);

int      rc  The return code.
NLProblem P  The problem.
int      c  Which constraint.
double   u  The upper bound.
```

### Description

This routine sets the upper bound on the inequality constraint. This can be queried with the NLPGetInequalityConstraintUpperBound (page 79) subroutine. The bounds can also be set at the same time using the NLPSetInequalityConstraintBounds (page 81) routine.

Initially the bound is  $\infty$ . (A value of  $1.e20$  is considered by Lancelot to be infinity.)

### Errors

Errors return 0 and make no changes to the problem. Normal execution returns 1.

Message	Severity
"Problem (argument 1) is NULL"	12
"Inequality constraint number %d (argument 2) is illegal. Must be in range 0 to %d"	12





## NLPGetNumberOfInequalityConstraints

### Purpose

Returns the number of inequality constraints in a problem.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
n=NLPGetNumberOfInequalityConstraints( $P$ );
```

int  $n$  The number of constraints.

NLProblem  $P$  The problem.



NLPEvaluateInequalityConstra()1(n)31(t)T050.446Td(Purp)31(ose)TF181.95Tf021.6

## NLPEvaluateGradientOfInequalityConstraint

### Purpose

Evaluates the gradient of an inequality constraint.

### Library

libNLPAPI.a

### C Syntax

NLPEvaluateHessianOfInequalityConstraint

## NLError

### Purpose

Queries whether an error condition exists.

### Library

`libNLPAPI.a`

### C Syntax

```
#include <NLPAPI.h>
rc
```

## NLGetNErrors

### Purpose

Returns the number of errors that have been flagged.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
n=NLGetNErrors();
```

int *n* The number of errors.

This routine returns the number of errors that have been set.



## NLGetErrorSev

## Purpose

Returns the severity of an error.

## Library

libNLPAPI.a

## C Syntax

```
#include <NLPAPI.h>
```

```
sev=NLGetErrorSev(i);
```

```
int sev The severity.
```

int /i(ns)ʒhɒlRɪt(ə)nLɪʒəThe 14.4.21 10330TdD1.955s542c(br)-(i)pt(i)onse





NLGetErrorLine

Purpose

## NLGetErrorFile

### Purpose

Returns the file containing the source code from which an error was issued.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
file=NLGetErrorFile(i);
```

char \**file*   The file.

int *i*       Which error.

## NLClearErrors

### Purpose

Clears all errors.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
void NLClearErrors();
```

### Description

This routine clears the error stack.

NLCreateVector

## NLCreateVectorWithSparseData

### Purpose

Allocates and initializes an NLVector data structure of a given length.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
v=NLCreateVectorWithSparseData(n, nz, el, v);
```

NLVector	<i>v</i>	The vector.
----------	----------	-------------

int	<i>n</i>	The length of the vector.
-----	----------	---------------------------

int	<i>nz</i>	The number of non-zeros in the vector.
-----	-----------	--

int	<i>*el</i>	The indices of the non-zero coordinates.
-----	------------	--

double	<i>*v</i>	The values of the non-zero coordinates.
--------	-----------	---

### Description

This routine, NLCreateVectorWithSparseData allocates and initializes an



Message	Severity
"Length of Vector %d (argument 1) is Illegal. Must be positive."	12
"Number of nonzeros in vector %d (argument 2) is Illegal. Must be nonnegative."	12
"The pointer to the array of nonZeros (argument 3) is NULL"	4
"The pointer to the array of coordinates (argument 4) is NULL"	4
"Out of memory, trying to allocate %d bytes"	12



Message

Severity

---

## NLCreateDenseWrappedVector

### Purpose

Allocates and initializes an NLVector data structure of a given length with

NLFreeVector

Purpose

## NLRefVector

### Purpose

Registers a reference to an NLVector data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
void NLRefVector(v);
    NLVector v The vector.
```

### Description

The NLVector structure is defined in nlpapi.h. It is used to register a reference to an NLVector data structure. The NLRefVector function registers a reference to an NLVector data structure. The NLRefVector function registers a reference to an NLVector data structure. The NLRefVector function registers a reference to an NLVector data structure.







edf  
alute i1Sfe coi1Sr(dinai1Ste)-317(fe)-317ae vecto.

~~duVGetC1Srx~~

---

## Purpose

Returns a coordinate of a vector.

## Library

libNLPAPI.a

~~Q Syntax~~  
a u y o f coi 1Sr(di nai 1Ste).

~~Wm~~1Studi 1Ste(NL)(LPA)1(PI)1(h>x)]T/35111. 95Tf0140. 457Td[cx



## NL CopyVect6r

---

### Purpose

Returns a copy of a vect6r.

### Library

libNLPU]T7I]T.a

### C Syntax

```
#include <NL]T7U]T7I]T.h>
```

```
v=NLCopyVectr( u
```

```
urReturns a(cop)27(y)498((6)1(f)499(as)498((v)28(cr)-1cC)1((r.-951]TJ7f)498((cop)
```





NLVInnerProd

Purpose







## NLVsparse

### Purpose

Queries if a vector is sparse.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
flag=NLVsparse(u);
```

int *flag* The answer. 1 indicates sparse, 0 dense

NLVector *u* The vector.

### Description

This routine returns 1 if the vector is a sparse vector (otherwise returns 0).

### Errors

Message

Severity

---

NLVnNonZeros

Purpose

## NLVnonZero

### Purpose

For a sparse vector returns a pointer to the array containing the list of nonzeros.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
c=NLVnonZero(u);
```

int\* *c* The list of which coordinates are nonzero.

NLVector *u* The vector.

**Description** `criptionf344a34326(0)1(2(ter)e)1(ro)34325943he344a(ns)26y260343326(44c)2`





## NLVGetNonZero

### Purpose

Returns the coordinate index of a non-zero coordinate in a vector.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
c=NLVGetNonZero(v, i);
```

double c The value of the *i*th non-zero coordinate.

NLVGetNonZero.

NLVWrapped

## NLVData

### Purpose

Returns a pointer to the data array of a vector.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
data=NLVData(u);
```

```
int
```



NLCreateMatrix

Purpose

## NLCreateMatrixWithData

### Purpose

Allocates and initializes an NLMatrix data structure of a given size with



## NLCreateSparseMatrix

### Purpose

Allocates and initializes an NLMatrix data structure of a given size.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
A=NLCreateSparseMatrix( $n$ ,  $m$ );
```

NLMatrix  $A$  The matrix.

int  $n$  The number of rows in the matrix.

int  $m$  The number of columns in the matrix.

### Description

The routine NLCreateSparseMatrix



Message	Severity
"Number of rows %d (argument 1) is negative."	12
"Out of memory, trying to allocate %d bytes"	12
"Out of memory, trying to allocate %dx%d matrix (%d bytes)"	12

## NLCreateDenseWrappedMatrix

### Purpose

Allocates and initializes a dense NLMatrix data structure of a given size, with a data array provided. If the user later changes the array the NLMatrix will see the changes.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
A=NLCreateDenseWrappedMatrix( n, m, data);
```

NLMatrix	<i>A</i>	The matrix.
int	<i>n</i>	The number of rows in the matrix.
int	<i>m</i>	The number of columns in the matrix.
double*	<i>data</i>	The data array.

### Description

The routine NLCreateDenseWrappedMatrix allocates and initializes an NLMatrix data structure of a given size. The matrix returned has the data array specified.

The NLMatrix data structure uses reference counting. The data structure should be deleted using the NLFreeMatrix subroutine (page 129). This will decrement the reference count and free the storage if the count goes to zero. References may be added using the NLRefMatrix subroutine (page 128).

### Errors

Errors return (NLMatrix)NULL.

Message

Severity

---

NLRefMatrix



NLFreeMatrix

Purpose



## NLMGetNumberOfCols

### Purpose

Returns the number of columns in an NLMatrix.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
m=NLMGetNumberOfCols(A);

int      m
```

## NLMGetElement

### Purpose

Returns an element of an NLMatrix.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
double aij = NLMGetElement(A, i, j);
```

double *aij*    The element of the matrix.

NLMatrix *A*    The matrix.

int *i*          The row index of the element.

int *j*          The column index of the element.

### Description

This routine returns the specified element of the matrix. This is set when the matrix is created, or with the NLMSetElement routine (page 133).

### Errors

Errors return DBL\_QNAN.

Message	Severity
"Matrix (argument 1) is NULL"	12

## NLMSetElement

### Purpose

Changes the value of an element of an NLMatrix.

### Library

libNLPAPI.a

### C Syntax

```
int NLMSetElement(NLMatrix *m, int row, int col, double value);
```

NLMIncrementElement

## NLMatrixDoubleProduct

### Purpose

Computes the product  $u^T A v$ .

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
p=NLMatrixDoubleProduct(u, A, v);
```

double  $p$  The return code.

NLVector  $u$  The vector operating on26(the)-327(pr)left.

NLMatrix  $A$  The matrix.

NLVector  $v$  The vector operating on26(the)-327(pr)right

**Description** This routine returns  $u^T A v$  if  $u$  is an  $m$ -v and  $v$  is an  $m$ -v and  $A$  is an  $m \times m$  matrix. If  $u$  is an  $m$ -v and  $v$  is an  $n$ -v and  $A$  is an  $m \times n$  matrix, the routine returns  $u^T A v$  if  $n = m$ . Otherwise, the routine returns an error message.

---

**NextMalt**

NVMVMuV

## Purpose

Computes the product  $b = Ax$ .

## Library

libc1xLPac1xPIc1x. a.

## Syc1xn

<NLc1xPac1xPIc1x. h>t



## NLMVMultT

### Purpose

Computes the product  $b = A^T x$ .

NLMSetToZero

Purpose



## NLGetGershgorinBounds

### Purpose

Computes bounds (using Gershgorin disks) of the leftmost eigenvalue of a matrix (with an optional diagonal scaling).

## NLMatrixOneNorm

### Purpose

Computes the 1-norm of a matrix (with an optional diagonal scaling).

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
L1=NLMatrixOneNorm(A, M);
double
```



NLM SumRankOneInto

Purpose







## NLMDEetermineHessianSparsityStructure

### Purpose

Updates the sparsity structure of a matrix to accomodate the nonzeros in the Hessian of the objective or a constraint of a problem.

### Library

## NLMData

### Purpose

Returns a pointer to the data array of the matrix.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
data=NLMData(A);
```

double\* *data*    The data array.

NLMMatrix *A*     The matrix.

### Description

This routine returns a pointer to the internal data array of the matrix.

NLMnE

### Purpose

Returns the number of “nonzero” entries in a matrix.

## NLMRow

### Purpose

Returns a pointer to the “row” array of the matrix.

### Library

libNLPAPI.a

### C Syntax

NLMCol

Purpose



## NLCreateGroupFunctionByString

### Purpose

Allocates and initializes an NLGroupFunction data structure by way of an expression.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
G=NLCreateGroupFunctionByString(P, type, var, expr);
```

NLGroupFunction *G*    The group function.

NLProblem *P*            The problem to which the group function belongs.

char \**type*



Message	Severity
"Pr31m4lem (1marg1mument 11m) is NULL"	11m2
"type (1marg1mument 21m) is NULL"	11m2
"v (arg1mument 31m) is NULL"	11m2
"expr (1marg1mument 41m) is NULL"	11m2



Message	Severity
"Out of memory, trying to allocate %d bytes"	12

NLRefGroupFunction

Purpose

## NLFreeGroupFunction

### Purpose

Frees the storage associated with an NLGroupFunction data structure.





## NLGEvalSecDer

### Purpose

Evaluates the second derivative of an NLGroupFunction.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
g=NLGEvalSecDer(G, x
```



## NLCreateElementFunctionByString

### Purpose

Allocates and initializes an NLElementFunction data structure by means of



## NLCreateElementFunction

### Purpose

Allocates and initializes an NLElementFunction data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```



## NLRefElementFunction

### Purpose

Registers a reference to an NLElementFunction data structure.

NLF5eeElementFunction

Purpose

## NLEGetDimension

### Purpose

Returns the number of unknowns (element internal variables) for an NLElementFunction.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
n=NLEGetDimension(F);

int          n   The number of unknowns.
NLElementFunction F The element function.
```

### Description

This routine returns the number of unknowns for an element function.

### Errors

Errors return -1.

Message	Severity
"Element Function (argument 1-21.6426(is)-327(NUL)1(L")-10772(1)1(2)]TJ.iL 167	







# NLEEvalSecDer

## Purpose

Evaluates the second derivative of an NLElementFunction.

## Library

libNLPAPI.a

## C Syntax

```
#include <NLPAPI.h>
f=NLEEvalSecDer(F, i, j, n, x);

double      f    The value of the second derivative.
NLElementFunction F  The element function.
int         i    The first variable.
int         j    The second variable.
int         n    The number of coordinates in the point.
double      *x   The point.
```

## Description

This routine returns the value of the second derivative of a element function  $d^2 f(x)/dx_i/dx_j$ .

## Errors

Errors return DBL\_QNAN.

Message	Severity
"Element Function (argument 1) is NUnL"	11(n2)]TJ0-14.446Td[("D)1(

NLCreateNonlinearElement

## NLRefNonlinearElement

### Purpose

Registers a reference to an NLNonlinearElement data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
void NLRefNonlinearElement( $P$ ,  $F$ );
```

        NLProblem                 $P$   The problem.  
        NLNonlinearElement       $F$   The element function.

### Description

The NLNonlinearElement data structure uses reference counting. This rou-

## NLFreeNonlinearElement

### Purpose

Frees the storage associated with an NLNonlinearElement data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
void NLFreeNonlinearElement( $P$ ,  $F$ );
```

NLProblem  $P$  The problem.

NLNonlinearElement  $F$  The element function.

### Description

The NLNonlinearElement data structure uses reference counting. This routine should be used to indicate that a vector is no longer needed. It will

## NLNEGetName

### Purpose

Returns the name of a nonlinear element.

### Library

libNLAPI.a

### C Syntax

```
#include <NLAPI.h>
name = NLNEGetName(
```









## NLNEGetIndex

### Purpose

Returns the index of an element variable of a nonlinear element.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
var NLNEGetIndex(P, ne, i);
```

int	<i>var</i>	The variable.
-----	------------	---------------

NLProblem	<i>P</i>	The problem.
-----------	----------	--------------

NLNEGetRangeXForm

## NLPGetNumberOfNonlinearElements

### Purpose

Returns the number of nonlinear elements.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
n=NLPGetNumberOfNonlinearElements(P);
```

int *n*            The number of nonlinear elements.

NLProblem *P*    The problem.

### Description

The routine NLPGetNumberOfNonlinearElements

## NLPGetNumberOfGroups

### Purpose

Returns the number of groups in a problem.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
n=NLPGetNumberOfGroups(P);

int      n    The number of groups.
NLProblem P  The problem.
```

### Description

This routine returns the current number of groups in a problem. Each time

## NLPGetTypeOfGroup

### Purpose

Returns the type of a group.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
type=NLPGetTypeOfGroup(P, i);
```

char	* <i>name</i>	The type of the group.
------	---------------	------------------------

NLProblem	<i>P</i>	The problem.
-----------	----------	--------------

int	<i>P</i>	
-----	----------	--

NLPGetGroupName

## NLPGetGroupName

### Purpose

Returns the name of a group.

### Library

libNLPAP#.a

### C Syntax

```
#include <NLPAP#.h>
```

```
name=NLPGetGroupName(P, i);
```

char	<i>*name</i>	The name of the group.
NLProblem	<i>P</i>	The problem.
int	<i>i</i>	The number of the group.

### Description

This routine returns the name of a group. Group names are assigned sequentially starting from 1.



## NLPSetGroupFunction

### Purpose

Sets the group function of a group.

## NLPGetGroupFunction

### Purpose

Gets the group function of a group.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
g=NLPGetGroupFunction(P, group
```

NLGroupFunction	<i>g</i>	The group function.
NLProblem	<i>P</i>	The problem.
int	<i>group</i>	The index of the group.

## NLP\_IsGroupFunctionSet

### Purpose

Queries whether the group function of a group has been set.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
ans
```



## NLPGetGroupA

### Purpose

Gets the linear part of the linear element of a group.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
a=NLPGetGroupA(P, group);
```

NLVector	<i>a</i>	The linear element.
----------	----------	---------------------

NLProblem	<i>P</i>	The problem.
-----------	----------	--------------

int	<i>group</i>	The index of the group.
-----	--------------	-------------------------

### Description

This routine sets the linear part of the linear element of a group. This can

be queried with the NLPGetGroupA (page 189) subroutine. The default value is 0.

## NLPIsGroupASet

### Purpose

Queries whether the linear part of the linear element of a group has been set.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
ans=NLPISGroupASet(P, group);
```















## NLPGetElementWeight

### Purpose

Returns the weight of a nonlinear element.

### Library

libNLPAPI.a

### C Syntax

## NLPSetElementWeight

### Purpose

Changes the weight of a nonlinear element.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
rc=NLPSetElementWeight(P, group, element, weight);
int
```



## NLPGetElementFunctionOfGroup

### Purpose

Returns the nonlinear element function of a nonlinear element.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
f=NLPGetElementFunction(P, group, element);
```

NLElementFunction	<i>f</i>	The element function.
NLProblem	<i>P</i>	The problem.
int	<i>group</i>	The index of the group.
int	<i>element</i>	The number of the nonlinear element.

### Description





## NLPGetElementFunction

### Purpose

Returns the nonlinear element function of a nonlinear element.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
f=NLPGetElementFunction(P
```

NLPSetElementFunction

Purpose

## NLPSetElementFunctionWithRange

### Purpose

Changes the nonlinear element function of a nonlinear element.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
rc=NLPSetElementFunctionWithRange(P, group, element, f, variables, xfrm);
```

int	<i>rc</i>	The return code.
NLProblem	<i>P</i>	The problem.
int	<i>group</i>	The index of the group.
int	<i>element</i>	The number of the nonlinear element.
NLElementFunction	<i>f</i>	The element function.
int	<i>it variables</i>	A list of the internal variables.
NLMatrix	<i>xfrm</i>	The range transformation.

### Description

This routine changes the nonlinear element function of an element of a group.

There must be as many entries in the list of ofvarialges s the et

transformation s many colmn(s)-279(as)-926(thr(e)-979()-1r(e)-979-289.11g)1(e)-1fr  
vaiaoblss many a-1(s)-327(the)-326(ele)-1(men)27(t)-326(function)1(n)-327h(a-1(s)-327unk

## NLPIsElementFunctionSet

### Purpose

Queries whether the weight of a nonlinear element has been set.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
ans=NLPISElementFunctionSet(P, group, element
```

NLPGetElementRangeTransformationOfGroup

Purpose

## NLPGetElementRangeTransformation

### Purpose

## **NLPGetNumberOfInternalVariablesInElement**

### **Purpose**

Returns the number of internal variables of a nonlinear element.

### **Library**











## NLPGetNumberOfElementsO

### Purpose

Returns the total number of nonlinear elements in the Objective.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
n=NLPGetNumberOfElementsi0(P);

int      n    The number of elements.
NLProblem P   The proP
```

### Description

This routine returns the total number of nonlinear elements in the Objective.

### Errors



NLPGetNumberOfElementsI

Purpose

## NLPGetElementTypeName

### Purpose

Returns the type name of a nonlinear element.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
name=NLPGetElementTypeName(P, group, element
```

### Errors

Errors return (char\*)EpEpNULL. T520611.581-21.297

"Element %d illegal (argument 3EpEp. Must beange



## NLPGetTypeOfElement

### Purpose

Returns the type name of a nonlinear element.

### Library

libNLPAPI.a



## NLPGetE6ementType

### Purpose

Returns the index of a type of nonlinear element.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
type=NLPGetEementType( P, i);
```

int            *type*    The index of the element type.

NLProblem    *P*        The problem.

int           *i*        Which type.

### Description

This routine returns the index of an element type. Element types are assigned with the NLCreateNonlinearElement (page 171) subroutine. A new type name is assigned a number, and the name is stored.

### Errors

Errors return -1.

Message	Severity
"Problem (argument 1) is NULL"	12

## NLPGetNumberOfGroupTypes

### Purpose

Returns the number of distinct types of groups.

### Library

libNLPAPI.a

### C Syntax

```
#include <L55GU7taligeGmb(nc)r0(ncfG(ncro)1(G)-1o)1(e1o)s(31(ta)35x
```



## NLCreateLancelot

### Purpose

Allocates and initializes an NLLancelot data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
Lan=NLCreateLancelot();

NLLancelot Lan The solver.
```

### Description

The routine NLCreateLancelot allocates and initializes an NLLancelot data structure. The solver returned has default parameters values, which can be set with various subroutines. Multiple instances are legal.

The storage used by the solver can be returned to the system using the NLFreeLancelot subroutine (page 224).

### Errors

Errors return (NLLancelot)NULL.

Message	Severity
"Out of memory, trying to allocate %d bytes"	12



## NLFreeLancelot

### Purpose

Releases storage associated with an NLLancelot data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
void NLFreeLancelot(Lan);

    NLLancelot Lan The solver.
```

### Description

The routine NLFreeLancelot returns storage associated with a solver to the system.

### Errors

Errors return without changing the solver.



## LNMinimize

### Purpose

Allocates and initializes an NLLancelot data structure.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
c
```



LNMaximize and LNMaximizeDLL

Purpose





## LNGetCheckDerivatives

### Purpose

Gets the parameter controlling how Lancelot test derivatives.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>  
flag=LNGnc.h>
```

## **LNSetConstraintAccuracy**

### **Purpose**

Sets the parameter controlling how accurately constraints are solved.

### **Library**

`libNLPAPI.a`

### **C Syntax**

```
#include <NLPAPI.h>
```





## **LNSetFirstConstraintAccuracy**

### **Purpose**

Sets the parameter controlling the initial accuracy Lancelot uses for the constraints.

### **Library**

`libNLPAPI.a`

### **C Syntax**

```
#include <NLPAPI.h>
```

LNGetFirstConstraintAccuracy

Purpose



## LNGetFirstGradientAccuracy

### Purpose

Gets the parameter controlling the initial accuracy for the gradients.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
acc=LNGetFirstGradientAccuracy(Lan);

double      acc    The accuracy.
NLLancelot  Lan    The solver.
```

### Description

The routine LNGetFirstGradientAccuracy gets the parameter controlling the initial accuracy for the gradients. The default value is 0.1. The

## LNSetGradientAccuracy

### Purpose

Sets the parameter controlling the accuracy for the gradients.

### Library

libNLPC

### C Syntax

```
#include <NL  
rc=LNSetGradientAccuracy( Lan, limit);  
  
int      rc      The return code.  
NLncelot Lan     The solver.  
double   limit   The accuracy.
```

### Description

The routine LNSetGradientCuracy

## LNGetGradientAccuracy

### Purpose

Gets the parameter controlling the accuracy for the gradients.

### Library



## LNGetInitialPenalty

### Purpose

Gets the parameter controlling the initial penalty.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>  
penalty
```





## **LNGetMaximumNumberOfIterations**

### **Purpose**

Gets the parameter controlling how long Lancelot runs.

### **Library**

libNLPAPI.a

LNSetPenaltyBound

## LNGetPenaltyBound

### Purpose

Gets the parameter controlling the bound on the penalty Lancelot uses.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>  
void penalty
```



## LNGetPrintEvery

### Purpose

Gets the parameter controlling how often Lancelot prints.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
iter=LNGetPrintEvery(Lan);

int      iter
NLLancelot Lan The solver.
```

### Description

The routine LNGetPrintEvery sets the parameter controlling how often Lancelot prints. The default value is 1.

Errors181.95Tf9.8.5594.445Td(Erro)1(rs)32(retur)1(n)32(1.)T11Message

---

LNSetPrintLevel

Purpose

## LNGetPrintLevel

### Purpose

Gets the parameter controlling how much output Lancelot produces.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
level=LNGetPrintLevel(Lan);

int          level
NLLancelot  Lan   The solver.
```

### Description

The routine LNGetPrintLevel



## LNSetPrintStart

### Purpose

Sets the parameter controlling when Lancelot starts printing.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>  
rc=LNSetPrintStart(Lan,
```

## LNGetPrintStart

### Purpose

Gets the parameter controlling when Lancelot starts printing.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
iter=LNGetPrintStart(Lan);

int          iter
NLLancelot  Lan  The solver.
```

### Description

The routine LNGetPrintStart sets the LTF911.95% controlling wrnclot;

## LNSetPrintStop

### Purpose

Sets the parameter controlling when Lancelot stops printing.



## LNSetRequireExactCauchyPoint

### Purpose

Sets the parameter determining whether an exact cauchy point is required.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
rc=LNSetRequireExactCauchyPoint(Lan, choice);

    int          rc          The return code.
    NLLancelot   Lan        The solver.
    int          choice
```

### Description

The routine LNSetRequireExactCauchyPoint sets the parameter determining whether an exact cauchy point is required. The default is 1.

The corresponding SPEC. SPC file entries are EXACT-CAUCHY-POINT-REQUIRED and INEXACT-CAUCHY-POINT-REQUIRED.

### Errors

Errors return 0, normal execution returns 1.

Message	Severity
"Solver (argument 1) is NULL"	12

LNGetRequireExactCauchyPoint

Purpose



## LNGetSaveDataEvery

### Purpose

Gets the parameter controlling how often Lancelot saves data.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
iter=LNGetSaveDataEvery(Lan);

int          iter
NLLancelot  Lan  The solver.
```

### Description

The routine LNGetSaveDataEvery





## **LNGetScalings**

### **Purpose**

Gets the parameter controlling how LEncelot uses scalings.

### **Library**

libNLPAPI.a

C SyntEx



## LNGetSolveBQPAccurately

### Purpose

Gets the parameter controlling the solution of the BQP.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
choice=LNGetSolveBQPAccurately(Lan);
```

```
int choice
```

```
LNGetSolveBQPAccurately(Lan,choice,LPD(e)sc)(br)#pt#
```

```
LNGetSolveBQPAccurately(Lan,choice,LPD(e)sc)(br)#pt#
```

## LNSetLinearSolverMethod

### Purpose

Sets the parameter determining what linear solver is used.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
rc=LNSetLinearSolverMethod(r1(So)t11.95Tf.152(So)73.47
```

"Modified MA27 preconditioned"  
MODIFIED-MA27-PRECONDITIONED-CG-SOLVER-USED

"Schnabel-Eskow preconditioned"  
SCHNABEL-ESKOW-PRECONDITIONED-CG-SOLVER-USED

"Users preconditioned"  
USERS-PRECONDITIONED-CG-SOLVER-USED

"Bandsolver preconditioned"  
BANDSOLVER-PRECONDITIONED-CG-SOLVER-USED

"Multifront"  
MULTIFRONT-SOLVER-USED

"Direct modified"  
DIRECT-MODIFIED-MULTIFRONTAL-SOLVER-USED

"CG method used"  
CG-METHOD-USED

## Errors

Errors return 0, normal execution returns 1.

Message	Severity
"Solver (argument 1) is NULL"	12
"Out of memory, trying to allocate %d bytes"	12
"Linear Solver Type \"%s\" (argument 2) is invalid"	12

## LNGetLinearSolverMethod

### Purpose

Gets the parameter determining what linear solver is used.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
int choice=LNGetLinearSolverMethod(Lan);

int          choice
NLLancelot   Lan    The solver.
```





ose

## LNSetStopOnBadDerivatives

### Purpose

Sets the parameter controlling how Lancelot deals with bad derivatives.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
rc=LNSetStopOnBadDerivatives(Lan, flag);

int          rc    The return code.
NLLancelot   Lan  The solver.
int          flag  What to do.
```

**Set 433** **Derivatives(**



## LNSetTrustRegionRadius

### Purpose

Sets the parameter controlling the radius of the trust region.

### Library

libburpoLPRusTI rus. aF1711. 95Tf0-21. 4Td[(Li )1(C(r)75(Syrusn)e)]tarusxF2111. 95Tf0-2

## **LNGetTrustRegionRadius**

### **Purpose**

Gets the parameter controlling the radius of the trust region.

### **Library**

libNLPR1cSIR1c.a.





## LNSetUseExactFirstDerivatives

### Purpose

Sets the parameter controlling how Lancelot gets derivatives.

### Library

libNLPAPI.a

### C Syntax

```
#include <NLPAPI.h>
```

```
rc=LNSetUseExactFirstDerivatives(Lan, flag);
```

int	<i>rc</i>	The return code.
-----	-----------	------------------

NLLancelot	<i>Lan</i>	The solver.
------------	------------	-------------

int	<i>flag</i>	What to do
-----	-------------	------------

### Description

The routine LNSetUseExactFirstDerivatives sets the parameter controlling how Lancelot gets derivatives. If flag is 0, differencing is used, otherwise exact derivatives are expected. The default value is 1 (exact derivatives). The SPEC.SPC entry this corresponds to







