Multifario (MF): Documentation

Michael E. Henderson
IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598

August 12, 2003

1 Introduction

Multifario (MF) is a package for continuing solution manifolds of nonlinear systems of equations. The algorithm is described in some detail in the paper:

> Henderson, M. E., "*Multiparameter Continuation: Computing Implicitly Defined Surfaces*", International Journal of Bifurcation and Chaos, Vol. 12, No. 3 (2002) pp. 451–476.

This document is a reference to the implementation that was used for the examples in that paper. An example, of a c8amped rod, has been submitted for publication –

1.1 Overview

1.2 Software Architecture

This code is what is sometimes called a framework. It does not attempt to provide "the best" iterative solver for solving nonlinear systems, or "the best" collection of inflated systems, singular point detection method or branch switching algorithm. These are all dependant on the particular problem being solved. Instead it provides a "Continuation Method", that can be used

MFAtlas Represents a collection of *charts*

2 Installation

The IMF's provided with the distribution require Lapack, and the makefile assumes that it is available in a library called liblapack.a . The blas will also be required. To install:

1. edit the file share/config.site to give local lib and include dirs where Linpack and Lapack can be found.

2. run the configure script "./configure"

3. create the libraries (installed in lib) "make"

4. create the utilities (installed in bin) "make utilities"

5. create the examples (installed in bin) "make examples"

ComputeSphere

ComputeSphereSub

Uses the MFTPBVP manifold, with $k = 1$. Demonstrates secondary bifurcation from a sequence of pitchfork bifurcations from a trivial branch.

PlotfileToPOV Creates a POV-Ray (available at www.povray.org) file from the plotfile. A sample .pov control file (genericPOVRay.pov) is included which sets up a camera and colors and renders a .pov file generated by PlotfileToPOV.

DrawAtlas Creates a Tiff-file (by choice if libtiff – www.libtiff.org is available) or a Postscript file (if it isn't) with a rendering of an atlas file.

DrawAtlasTS Creates a Tiff or Postscript file with a rendering of an atlas file (with charts drawn in the tangent spaces, faster than DrawAtlas).

DrawDual Creates a Tiff or Postscript file with a rendering of the dual triangulation of the atlas file.

DualToDX Creates a DataExplorer file with the dual triangulation of the atlas file.

To create an image of the sphere run "bin/DrawPlotfile Sphere". This looks in the current directory for aatlascalledat134Sphere.view"awhich contains

Figure 1: The output from the ComputeSphere example.

```
MFImplicitMF M;
```

```
  MFFreeNVector(u0);
  MFFreeHe-dersonsMethod(H);

  return 0;
 }
```

Of course, whe- mai- ends all the storage is free'd anyway, but this is a good habit.

6 Example – solving a two point boundary value problem, the MFTPBVP manifold

Below we dissect the ComputeDomokos example, which should be enough to get you started with the TPBVP solver. The ComputeRod is more realistic, but the solution has sheets with symmetries, and the example separates these by controlling crossings of the planes of symmetry. The problem is from the paper

> Domokos, G. "Global Description of Elastic Bars". ZAMM – Z. angew. Math. Mech. **74** (1994) 4, T 289–T291.

```
int MFDomokosProjectToDraw(MFNVector,double*,void*);
void MFTPBVPSetStability(MFImplicitMF,MFNVector,MFNKMatrix,
                                                      void*);
int MFStopTPBVP(MFImplicitMF,MFNVector,MFNKMatrix,MFNVector,
                                             MFNKMatrix,void*);

#define PI 3.14159265358979323846264338327950288

#define NX 100

int main(int argc, char *argv[])
 {
```

boundary value problem. This is of the form

$$u' = f(t, u, p, u0, p0)$$
$$a(u(0), u(1), p, u0(0), u0(1), p0) = 0$$
$$\int_0^1 l(t, u(t), p, u0(t), p0)\,\mathrm{d}t + m(p, p0) = 0$$

The routines are passed as triples (except for m), of the routine to evaluate the function, and it's derivatives w.r.t. u and the parameters p. The pair $u0$ and $p0$ are a nearby function and parameter (for imposing phase constraints).

them all.) Fnally in line 62 we give a prefix for files (e.g. the plotfile will be Domokos.plotfile).

```
53  H=MFCreateHendersonsMethod();
```

Of course, when main ends all the storage is free'd anyway, but this is a good habit.

Finally, there are the routines defining the problem and the projection.

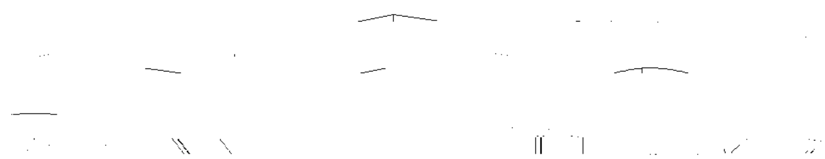

Figure 2: The output from the ComputeDomokos example.

What the implementations below are missing (and AUTO and LOCA prodouble MFIMFScale(M,u,Phi); – Returns a radius for the ball at a point *u*, at which the columns of the matrix Phi give an o.n. basis for the tangent space. The idea is that for points in the tangent space that

11 MFNVector – a point in the embedding space

These are points lying in the embedding space. Again, the interface is quite a bit simpler than the IMF.

MFNVector MFCreateNVector(int); – Creates and returns an N vector of the given length. This ctor creates a vector stored as an array of doubles. It should be Free'd with the MFFreeNVector routine when it is no longer needed.

MFNVector MFCreateNVectorWithData(int,double*); – Creates and returns an N vector of the given length, with coordinates copied from the array. This ctor creates a vector stored as an array of doubles. It should be Free'd with the MFFreeNVector routine when it is no longer needed.

MFNVector MFCnector(MFNVector u); – Creates and returns an N vector of the same length and coordinates as u. Note: this is a "deep" copy, so changing a coordinate of the cnd vector does not change the corresponding coordinate of the original. This is a ctor, and the vector should be Free'd with the MFFreeNVector routine when it is no longer needed.

int MFNV_NC(MFNVector); – Returns the number of coordinareturnsof an NVector (i.e. *n*).

double MFNV_C(MFNVector,int); – Returns the specified coordinate of an NVector.

void MFNVSetC(MFNVector,int,double); – Changes the specified coordinate of an NVector.

void MFNVAdd(a,b,c); – Adds two NVectors c=a+b. c must have been void MFFreeNVector(MFNVector); – Release a reference to the NVector. When the reference count goes to zero the storage associated with the object is free'd.

12 MFKVector – a point in the tangent space

These are points lying in the tangent space of the manifold. They are stored as a vector of doubles. The user normally would not need to use these objects.

MFKVector MFCreateKVector(int); – Creates and returns an K vector of the given length. This ctor creates a vector stored as an array of dou-

int MFNKMatrixN(MFNKMatrix); – Returns the length of the columns in the matrix.

MFNVector MFMColumn(MFNKMatrix,int); – eturns the requested column.

void MFNKMSetC(MFNKMatrix,int i,int j,double); – Changes the *i*th element of *j*th column to the given value.

MFPolytope MFChartPolytope(MFChart); – Returns the Polyhedron associated with a chart.

MFNVector MFChartCenter(MFChart); – Returns the center of a chart.

MFNKMatrix MFChartTangentSpace(MFChart); – Returns an o.n. basis for the tangent space of the manifold at the center of a chart.

double MFChartRadius(MFChart); – Returns the radius of a chart.

int MFChartEvaluate(MFChart,MFKVector s,MFNVector u); – Projects a point in the domain of the chart onto the manifold. The NVector u must have been allocated by the user and should be the same type as the chart center.

int MFChartInterior(MFChart,MFKVector); – Tests to see if a point is interior to the polyhedron of a chart.

int MFChartHasBoundary(MFChart); – Tests to see if all vertices of the polyhedron of a chart have radius less than the radius of the chart.

int MFChartK(MFChart); – ReturnsThe2-6dimension ofheld a chart.

int MFChartN(MFChart); – Returns the dimension of the embedding space of a chart.

void MFChartProjectIntoTangentSpace(MFChart,MFNVector u,MFKVector s); – Projects a point in 8he embedding space orthogonally onto 8he tangent

15 MFContinuationMethod – an algorithm for computing an atlas of charts for a manifold

void MFHendersonsMethodSetMinR(H,int); – Sets the minimum chart radius. Must be positive. A chart wi radius below this limit is treated as if it is interior. The current setting can be retreived using the routine MFHendersonsMethodGetMinR.

NOTE: I'm still working on this.

void MFHendersonsMethodSetDumpToRestartFileEvery(H,int); – Indicates how dense the points in the restart file are. As charts are added each is assigned a number, which is the minimum of the numbers assigned to its neighbors, plus one. This is a rough indication of how many int MFAtlasAddChartWithAll(A,u,Phi,double R); – Adds a chart centered at the given point u with tangent space Phi, and radius R to the atlas.

MFImplicitMF MFAtlasMF(A); – Returns the manifold corresponding to the atlas.

int MFAtlasNumberOfCharts(A); – Returns the number of charts in the atlas.

17 Error handling

These routines provide a way of finding out what errors have occured in the

void MFNVectorSetSetC(MFNVector,void (*)(int,double,void*));– Sets the routine used to change a coordinate. This is meant to be a fallback in case a routine has to deal with a vector of unknown type. (Performance suffers if used to access long vectors.)

void MFNVectorSetAdd(MFNVector,void (*)(void*,void*,void*)); – Sets the routine used to add two vectors.

void MFNVectorSetDiff(MFNVector,void (*)(void*,void*,void*)); – Sets the routine used to multiply a vector by a scalar.

void MFNVectorSetPrint(MFNVector,void (*)(FILE*,void*)); – Sets the routine used to print a readable version of a vector.

19 Implementing an MFNSpace

void MFNSpaceSetFreeData(MFNSpace,void (*freedata)(void *)); – Sets the routine that is called when the last reference to the vector is Free'd. Note that the CreateBaseClass returns a vector with one reference.

void *MFNVectorGetData(MFNVector); – returns the data pointer of a vector.

void MFNSpaceSetDistance(MFNSpace,distance); – Sets the routine that computes the distance between two vectors in the space. The routine has the signature:

```
the distance
0.2]Td[(v)27.77402]To77402],Mo 78nc
```

20 Implement7ng an MFNRegion

The MFNRegion represents a subset of an *n*-dimensional space. The only real function is supplies is a ”test”.

The data pointer is passed as the last argument. If u or the Pu is passed as NULL the routine is expected to return the required length

```
void MFIMFSetStop(M,stop);
```