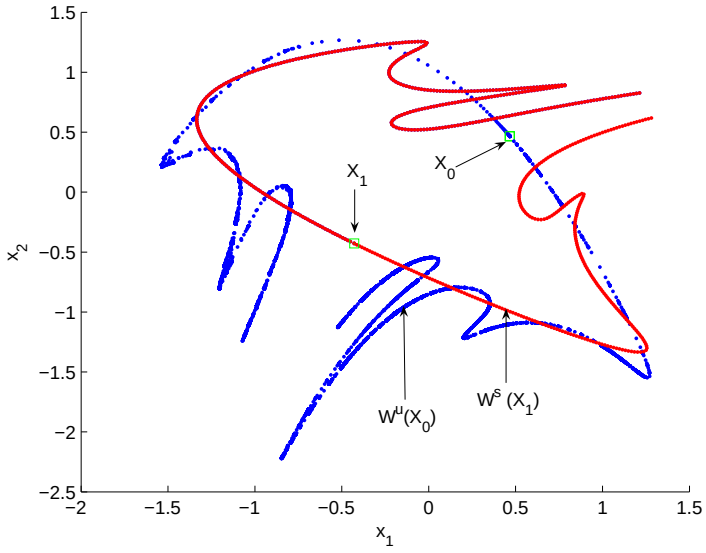
MatContM:
A toolbox for continuation and bifurcation of cycles of maps: Command line use.

H.G.E. Meijer, W. Govaerts, Yu. A. Kuznetsov
R. Khoshsiar Ghaziani, N. Neirynck

November 14, 2017



Belgium

Utrecht University
The Netherlands

UNIVERSITY OF TWENTE. The Netherlands

Contents

1 Introduction

The present manual on MatContM is based on version 5.3 of MatContM and the runs were tested on Matlab 9.3 (R2017b). It is meant to be a practical guide for MatContM users without undue discussion of the details of the used algorithms. Parts of this manual can be used as tutorials since all discussed command line runs are also available in the directory `Testruns` of the distributed version 5.3 of MatContM. In this case one may start with §2.6 and continue with §10.1. There are separate tutorials for the GUI version of MatContM.

In a typical use of MatContM, one starts with an initial fixed point or cycle, which may be obtained from analysis, simulations or previous continuations. One first computes curves of fixed points or cycles under variation of one parameter, and may detect bifurcation points on such curves. Starting from such bifurcation points, the continuer algorithm in MatContM can compute bifurcation curves. These curves are defined by a system of equations consisting of fixed point and bifurcation conditions. With one free parameter we can also compute curves of connecting orbits. By varying two system parameters we can compute bifurcation curves of limit points, period-doubling and Neimark-Sacker points as well as tangencies of homoclinic and heteroclinic orbits. A recent application is given in [45].

The following list contains functionalities that are provided by MatContM:

- Simulation (iteration) of maps, i.e. computation and visualization of orbits (trajectories).
- Computation of the Lyapunov exponents of long trajectories.
- Continuation of fixed points of maps and iterates of maps with respect to a control parameter.
- Detection of fold (limit point), flip (period-doubling point), Neimark-Sacker and branch points on curves of fixed points.
- Computation of normal form coefficients for fold, flip and Neimark-Sacker bifurcations.
- Continuation of fold, flip and Neimark-Sacker bifurcations in two control parameters.
- Detection of all codimension 2 fixed point bifurcations on curves of fold, flip and Neimark-Sacker bifurcations.
- Computation of normal form coefficients for all codimension 2 bifurcations of fixed points.
- Switching to the period doubled branch in a flip point.
- Branch switching at branch points of fixed points.
- Switching to branches of codimension 1 bifurcations rooted in codimension 2 points.
- Automatic differentiation for normal form coefficients of codimension 1 and codimension 2 bifurcations.
- Computation of one-dimensional invariant manifolds (stable and unstable) and in the two-dimensional case computing their transversal intersections to obtain initial homoclinic and heteroclinic connections.

- Continuation of homoclinic and heteroclinic orbits with respect to a control parameter and the detection of tangencies on the curve of orbits.
- Detection of other bifurcations on curves of homoclinic and heteroclinic orbits, see [46] for details.
- Continuation of homoclinic and heteroclinic tangencies in two control parameters.
- Detection of generalized tangencies and other bifurcations on curves of homoclinic and heteroclinic orbits, see [46] for details.

2 Basic aspects of numerical continuation and the software

2.1 Numerical continuation

In general, numerical continuation methods are used to compute solution manifolds of non-linear systems of the form:

$$F(X) = 0, \tag{1}$$

where $X \in \mathbb{R}^{n+k}$ and $F : \mathbb{R}^{n+k} \to \mathbb{R}^n$ is a sufficiently smooth function. The solutions of this equation consist of regular pieces, which are joined at singular solutions. The regular pieces are curves when $k = 1$, surfaces when $k = 2$ and k-manifolds in general.
In MatContM we use numerical continuation methods for analyzing the solutions of (1) when restricted to the case $k = 1$. In fact, we construct solution curves Γ in

$$\{X : F(X) = 0\}, \tag{2}$$

by generating sequences of points $X_i, i = 1, 2, ...$ along the solution curve Γ satisfying a chosen tolerance criterion. The general idea of a continuation method is that of a predictor-corrector scheme. Starting with an initial point on the continuation path, the goal is to trace the remainder of the path in steps. At each step, the algorithm first predicts the next point on the path, and subsequently corrects the predicted point towards the solution curve. A variant of Newton's method is used for the corrector step. For details of the continuation method used in MatContM, we refer to [12, 13].

2.2 Test functions for bifurcations

Let $X = X(s)$ be a smooth, local parameterization of a solution curve of (1) where $k = 1$. Suppose that $s = s_0$ corresponds to a bifurcation point. A smooth scalar function $\psi : \mathbb{R}^{n+1} \to \mathbb{R}^1$ defined along the curve is called a test function, a tool to detect singularities on a solution branch, for the corresponding bifurcation if $g(s_0) = 0$, where $g(s) = \psi(X(s))$. The test function ψ has a regular zero at s_0 if $\frac{dg}{ds}(s_0) \neq 0$. A bifurcation point is detected between two successive points X_0 and X_1 on the curve if $\psi(X_0)\psi(X_1) < 0$. To solve the system

$$\begin{cases} F(X) = 0 \\ \psi(X) = 0 \end{cases} \tag{3}$$

we use a one-dimensional secant method to locate $\psi(X) = 0$ along the curve. Notice that this involves Newton corrections at each intermediate point.

2.3 Singularity matrix

Suppose we have two singularities S_1 and S_2, and test functions ψ_1 and ψ_2. Assume that ψ_1 vanishes at both S_1 and S_2 while ψ_2 generically vanishes only at S_2. Then we need to require that ψ_2 does not vanish at S_1, i.e. we need the possibility to require that in some singularities certain test functions do not vanish. To represent all singularities we use a singularity matrix, i.e. a compact way to describe the relation between the singularities and the test functions. Suppose we have n_s singularities and n_t test functions. Then the singularity matrix S is an $n_s \times n_t$ matrix, such that:

$$S_{ij} = \begin{cases} 0 & \text{means : for singularity i testfunction j must vanish} \\ 1 & \text{means : for singularity i testfunction j must not vanish} \\ otherwise & \text{means : for singularity i ignore test function j} \end{cases} \tag{4}$$

2.4 User functions

The user has the possibility to define specific functions which must be scalar and can depend only on the state variables and parameters. He can request that their zeros are detected and computed during continuation runs as if they were singular points. This requires that the options `Userfunctions` and `UserfunctionsInfo` are set properly (see §2.5.3) and that the *mapfile* defines the user functions (§2.6). An example application is given in §10.1.

2.5 Software

2.5.1 Continuer

The syntax of the continuer is:

$$[\mathtt{x}, \mathtt{v}, \mathtt{s}, \mathtt{h}, \mathtt{f}] = \mathtt{cont}(@\mathtt{curve}, \mathtt{x0}, \mathtt{v0}, \mathtt{options})$$

`curve` is a MATLAB m-file where the problem is specified, cf. §2.5.2.
`x0` and `v0` are respectively the initial point and the tangent vector at the initial point where the continuation starts.
`options` is a structure as described in §2.5.3.
The function returns:
`x` and `v`, i.e. the points and their tangent vectors along the curve. Each column in x and v corresponds to a point on the curve.
`s` is an array whose structures contain information on detected singularities. This structure has the following fields:

s.index	index of the singularity point in x
s.label	label of the singularity
s.data	any kind of extra information
s.msg	a string containing a message for this particular singularity

`h` is used for output of the algorithm, currently this is a matrix with for each point a column with the following components (in that order) :

- Stepsize:
  Stepsize used to calculate this point (zero for initial point and singular points)
- Half the number of correction iterations, rounded up to the next integer
  For singular points this is the number of locator iterations

- User function values :
  The values of all active user functions
- Test function values :
  The values of all active test functions

In general, `f` can be anything depending on which curve file is used. However, in MATCONTM, `f` always contains the multipliers if they were computed during the continuation. Multipliers are computed when *options* is set by :

```
options=contset(options,'Multipliers',1);
```

See §2.5.3 for more details.

It is also possible to extend the most recently computed curve with the same options (also the same number of points) as it was first computed. The syntax to extend this curve is:

```
[x, v, s, h, f] = cont( x, v, s, h, f, cds)
```

`x`, `v`, `s`, `h` and `f` are the results of the previous call to the continuer and *cds* is the global variable that contains the curve description of the most recently computed curve. The function returns the same output as before extended with the new results.

2.5.2 Curve file

The continuer uses special m-files in which the type of the solution branch is defined.
The basic four curve files (but not the only ones) in MATCONTM are `fixedpointmap.m`, `limitpointmap.m`,
`perioddoublingmap.m` and `neimarksackermap.m` in which defining systems for fold, flip and Neimark-Sacker bifurcations of cycles of maps are defined, respectively.

A curve file contains some sections as *curve_func*, *jacobian*, *hessians*, *adapt*, etc. In some cases the problem definition uses auxiliary entities like bordering vectors and it may be needed to adapt them during the continuation. In *adapt* these entities are adapted. If *cds.options.Adapt* has a value n, then after n computed points a call to
`[reeval,x,v]=feval(cds.curve_adapt,x,v)` will be made.

2.5.3 Options

In the continuation we use the *options* structure which is initially created with *contset*:
`options = contset`
will initialize the structure. The continuer stores the handle to the options in the variable *cds.options*. Options can then be set using
`options = contset(options, optionname, optionvalue);`
where *optionname* is an option from the following list.

InitStepsize the initial stepsize (default: 0.01)

MinStepsize the minimum stepsize to compute the next point on the curve (default: 10^{-5}).
It is implicitly assumed that the minimum stepsize is not larger than the initial stepsize but this is not checked by the code.

MaxStepsize the maximum stepsize (default: 0.1). It is implicitly assumed that the maximum stepsize is not smaller than the initial stepsize but this is not checked by the code.

MaxCorrIters maximum number of correction iterations (default: 10)

MaxNewtonIters maximum number of Newton-Raphson iterations before switching to Newton-Chords in the corrector iterations (default: 3)

MaxTestIters maximum number of iterations to locate a zero of a test function (default: 10)

Increment the increment to compute the derivatives numerically (default: 10^{-5})

FunTolerance tolerance of function values: $||F(x)|| \leq FunTolerance$ is the first convergence criterion of the Newton iteration (default: 10^{-6})

VarTolerance tolerance of coordinates: $||\delta x|| \leq VarTolerance$ is the second convergence criterion of the Newton iteration (default: 10^{-6})

TestTolerance tolerance of test functions (default: 10^{-5})

Singularities boolean indicating the presence of singularities (default: 0)

MaxNumPoints maximum number of points on the curve (default: 300)

Backward boolean indicating the direction of the continuation (direction of the initial tangent vector) v_0 (default: 0)

CheckClosed number of points indicating when to start to check if the curve is closed (0 = do not check) (default: 50)

Adapt number of points indicating when to adapt the problem while computing the curve (0 = do not adapt) (default: 3)

IgnoreSingularity vector containing indices of singularities which are to be ignored (default: empty)

Multipliers boolean indicating the computation of the multipliers (default: 0)

TSearchOrder numerical value that indicates if unit vectors are cycled in increasing order of index (default: 1, increasing) or decreasing (set to a value different from 1), see §2.5.12.

Userfunctions boolean indicating the presence of user functions (default: 0)

UserfunctionsInfo is an array with structures containing information about the user functions. This structure has the following fields:

.label	label of the user function (must consist of four characters, including possibly trailing spaces)
.name	name of this particular user function
.state	boolean indicating whether the user function has to be evaluated or not

For the options MaxCorrIters, MaxNewtonIters, MaxTestIters, Increment, FunTolerance, VarTolerance, TestTolerance and Adapt the default values are in most cases good.

Options also contains some fields which are not set by the user but frozen or filled by calls to the curvefile, namely:

MoorePenrose boolean indicating the use of the Moore-Penrose continuation as the Newton-like corrector procedure (default: 1)

SymDerivative the highest order symbolic derivative which is present (default: 0)

SymDerivativeP the highest order symbolic derivative with respect to the parameter(s) which is present (default: 0)

AutDerivative boolean indicating the use of automatic differentiation in the computation of normal form coefficients (default: 1)

AutDerivativeIte an integer number that indicates the use of automatic differentiation when the iteration number of the map equals or exceeds this number (default: 24)

Testfunctions boolean indicating the presence of test functions and singularity matrix (default: 0)

WorkSpace boolean indicating to initialize and clean up user variable space (default: 0)

Locators boolean vector indicating for which testfunctions a specific locator code exists to locate its zeroes. Otherwise the default locator is used (default: empty)

ActiveParams vector containing indices of the active parameter(s) (default: empty)

Some more details follow here on some of the options.

2.5.4 Derivatives of the defining system of the curve

In the defining system of the object that is to be continued, the derivates can be provided that are needed for the continuation algorithm or other computations. The continuer stores the handle to the derivatives in the variables `cds.curve_jacobian`,`cds.curve_hessians`.

If `cds.symjac`= 1, then a call to `feval(cds.curve_jacobian, x)` must return the $(n-1) \times n$ Jacobian matrix evaluated at point x.

If `cds.symhess`= 1, then a call to `feval(cds.curve_hessians, x)` must return a 3-dimensional $(n-1 \times n \times n)$ matrix H such that $H(i,j,k) = \frac{\partial^2 F_i(x)}{\partial x_j \partial x_k}$.

In the present implementation in most cases `cds.symhess`= 0, so the ODE-file does not provide second order derivatives, since they are not needed in the algorithms used.

2.5.5 Singularities and test functions

To detect singularities on the curve one must set the option *Singularities* on. Singularities are defined using the singularity matrix, as described in section 2.3. The continuer stores the handles to the singularities, the testfunctions and the processing of the singularities respectively in the variables `cds.curve_singmat`,`cds.curve_testf` and `cds.curve_process`.

A call to `[S,L] = feval(cds.curve_singmat)` gets the singularity matrix S and a vector of 2-character strings which are abbreviations of the singularities.

A call to `feval(cds.curve_testf, ids, x, v)` then must return the evaluation of all testfunctions, whose indices are in the integer vector `ids`, at `x` (`v` is the tangent vector at `x`). As a second return argument it should return an array of all testfunction id's which could not be evaluated. If this array is not empty the stepsize will be decreased.

When a singularity is found, a call to `[failed,s] = feval(cds.curve_process,i,x,v,s)` will be made to process singularity `i` at `x`. This is the point where computations can be done, like computing normal forms, eigenvalues, etc. of the singularity. These results can then be saved in the structure `s.data` which can be reused for further analysis. Note that the first and last point of the curve are also treated as singular.

2.5.6 Locators

It may be useful to have a specific locator code for locating certain singularities. To use a specific locator you must set the option *Locators*. This is a vector in which the index of an element corresponds to the index of a singularity. Setting the entry to 1 means the presence of a user-defined locator. The continuer has stored the handles to the locators in the variable `cds.curve_locator` and will then make a call to
`[x,v]=feval(cds.curve_locate,i,x1,v1,x2,v2)`
to locate singularity i which was detected between `x1` and `x2` with their corresponding tangent vectors `v1` and `v2`. It must return the located point and the tangent vector at that point. If the locator was unable to find a point it should return `x = []`.

2.5.7 User functions

To detect zeros of userfunctions on the curve one must set the option *Userfunctions* on. The continuer has stored the handles to the userfunctions `cds.curve_userf`. First a call to `UserInfo = contget(cds.options, 'UserfunctionsInfo', [])` is made to get information on the userfunctions. A call to `feval(cds.curve_userf, UserInfo, ids, x, v)` then must return the evaluation of all userfunctions ids, whose information is in the structure `UserInfo`, at `x` (`v` is the tangent vector at `x`). As a second return argument it should return an array of all user function id's which could not be evaluated. If this array is not empty the stepsize will be decreased.

A special point on a bifurcation curve that is specified by a user function has a structure as follows:

`s.index`	index of the detected singular point defined by the user function.
`s.label`	a string that is in `UserInfo.label`, label of the singularity.
`s.msg`	a string that is set in `UserInfo.name`.
`s.data`	an empty tangent vector or values of the user functions in the singular point.

When a change of sign of a userfunction is detected, the userfunction `i` is processed at `x`. This is the point where the results (values of the userfunction) can be saved in the structure `s.data` which can be reused for further analysis.

2.5.8 Defaultprocessor

In many cases it is useful to do some general computations for every calculated point on the curve. The results of these computations can then be used by for example the testfunctions. The continuer has stored the handle to the defaultprocessor in the variable `cds.curve_defaultprocessor`.

The defaultprocessor is called as
`[failed,f,s] = feval(cds.curve_defaultprocressor,x,v,s)`.
`x` and `v` are the point on the curve and it's tangent vector. The argument `s` is only supplied if the point is a singular point, in that case the defaultprocessor may also add some data to the `s.data` field. If for some reason the default processor fails it should set `failed` to 1. This will result in a reduction of the stepsize and a retry which should solve the problem. Any information that is to be preserved, should be put in `f`. `f` must be a column vector and must be of equal size for every call to the default processor.

2.5.9 Special processors

After a singular point has been detected and located a singular point data structure will be created and initialized. If there are some special data (like eigenvalues) which may be of interest for a particular singular point then a call to `[failed,s] = feval(cds.curve_process,i,x,v,s)` should store this data in the `s.data` field. Here `i` indicates which singularity was detected and `x` and `v` are the point and tangent vector where this singularity was detected.

2.5.10 Workspace

During the computation of a curve it is sometimes necessary to introduce variables and do additional computations that are common to all points of the curve. The continuer has stored the handle to the initialization and cleaning of the workspace in the variables `cds.curve_init` and `cds.curve_done`. These can be relegated to a call of the type

```
feval(cds.curve_init,x,v).
```

This option has to be provided only if the variable `WorkSpace` in `cds.options` is switched on. In this case a call

```
feval(cds.curve_done,x,v)
```

must clear the workspace. Variables in the workspace must be set global.

2.5.11 Adaptation

It is possible to adapt the problem while generating the curve. If *Adapt* has a value, say 5, then after 5 computed points a call to `[reeval,x,v]=feval(cds.curve_adapt,x,v)` will be made where the user can program to change the system.

For some applications it is useful to change or modify the used test functions while computing the curve (like in bordering techniques). In order to preserve the correct signs of the test functions it is sometimes necessary to reevaluate the test functions after adaptation. To do this `reeval` should be one otherwise zero. The return variables `x` and `v` should be the updated `x` and `v` which may have changed because of the changes made to the system.

2.5.12 Tangent search order

To start a continuation, an initial point x_0 and a tangent vector v_0 are needed in general. Often, only x_0 is available. In this case, MatContM successively tries all unit vectors as candidate tangent vectors. By default, this is done in increasing order of index

(cds.options.TSearchOrder = 1). If cds.options.TsearchOrder is set to a value different from 1 then the cycling is done in decreasing order of index.

In cases where the number of continuation variables is large (e.g. when computing homoclinic connections) the choice of cds.options.TSearchOrder can substantially change the speed of the computation.

2.5.13 Directories

To start MatContM its main directory must be the root directory of MatcontM where `init.m` is set. The files of the toolbox are organized in the following subdirectories

- Continuer
  Here are all the main files for the continuer which are needed to calculate and plot any curve.
- FixedPointMap
  Here are all files needed to do a continuation of fixed points of iterates of a map. This includes in particular the initializers and the fixed point curve definition file.
- LimitPointMap
  Here are all files needed to do a fold continuation. This includes in particular the initializers and the fold curve definition file.
- PeriodDoublingMap
  Here are all files needed to do a flip continuation. This includes in particular the initializers and the flip curve definition file.
- NeimarkSackerMap
  Here are all files needed to do a Neimark-Sacker continuation. This includes in particular the initializers and the Neimark-Sacker point curve definition file.
- MultilinearForms
  Here are all files needed to compute the critical normal form coefficients for all codim-1 and codim-2 bifurcation points both numerically with finite directional differences and using symbolic derivatives of the original map.
- AD
  Contains all files needed to use automatic differentiation in the computation of multilinear forms.
- InvManifold
  Contains all files needed for the computation of stable and unstable manifolds.
- Homoclinic
  Here are all files needed to continue a homoclinic connection.
- Heteroclinic
  Contains all files needed to continue a heteroclinic connection.
- HomoclinicT
  Here are all files needed to continue a fold curve of homoclinic connections.

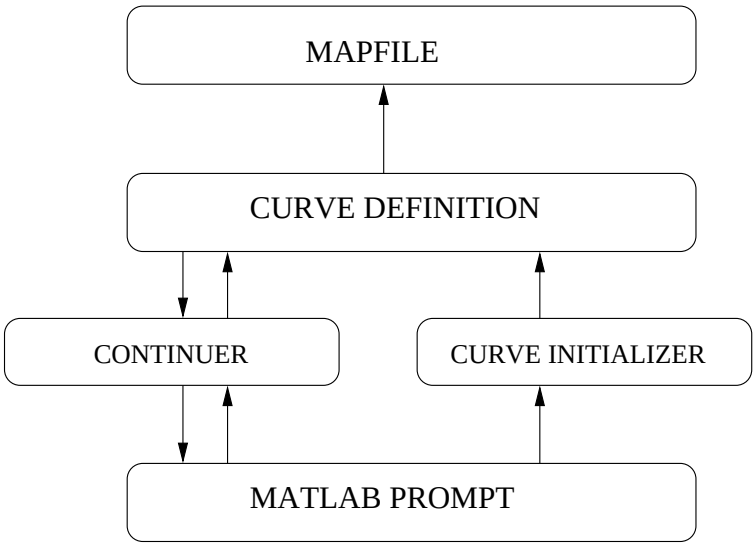


Figure 1: Continuation process in MatContM.

- HeteroclinicT
  Here are all files needed to continue a fold curve of heteroclinic connections.
- Systems
  Here are all example system definitions.
- Testruns
  Here are all example testruns. They can be used to run the examples described in this manual and to test if everything is working correctly. This directory contains three subdirectories *Tnfmap*, *LeslieGower* and *CodStock* that correspond to the testruns of the examples in §10.1, §10.2 and §10.3 respectively.
- GUI
  Here are the files which are specific to the GUI version of MatContM. These will not be discussed in the present manual.

The only files which are not in any of these directories are `init.m`, `cpl.m`, and `matcontm.m`. The command `init` must be called before any continuation in the toolbox, so that MATLAB can find all the needed functions. The function `cpl` is used to plot the results obtained in a continuation run. It can provide 2D or 3D plots. For instance the commands `cpl(x24,v24,s24,[3 1])` and `cpl(x4,v4,s4,[3 1 2])` create the 2D and 3D plots of Figure 10 and Figure 8, respectively. The command `matcontm` starts the GUI of MatContM and will not be discussed further in this manual.
The process of continuation is visualized in Figure 1.

2.6 The mapfile of the map

A solution curve must be initialized before doing a continuation. Each curve file has its own initializers which use a *mapfile* where the map is defined, see for instance § 3.2. A *mapfile* contains at least the following sections (in that order):

init, *fun_eval*, *jacobian*, *jacobianp*, *hessians*, *hessiansp*, *der3*, *der4* , *der5*.

However, the section *init* is routinely empty; it is reserved for users with special desiderata. A *mapfile* may also contain one or more sections that describe user functions.

We note that if state variables, parameters or user functions are added or deleted then this constitutes another dynamical system. So either all computed data should be deleted or ignored, or the name of the system should be changed. The last option is recommended, in particular when the GUI is used.

A *mapfile* can be defined by simply using the MATLAB editor or any other text editor.

From version 4.1 on MATCONTM provides two better possibilities to create mapfiles. One of them is by using the GUI version of MATCONTM. The other one is a shortcut to this.

In order to illustrate this method and the elements of a *mapfile*, we consider the example M_{TN}, the map of a truncated normal form that will be studied in Section 10.1.1.

A new *mapfile* can be created by calling `SysGUI.new`.

This opens a **System** window, which contains several fields and buttons. To identify the system, type for example

```
Tnfmap
```

in the **Name** field (it must be one word).

Input names of the **Coordinates**: `ksi1,ksi2`, and the **Parameters**: `beta1,beta2,CC,DD`.

If shown, select symbolic generation of the 1st order derivatives by pressing the corresponding radio-button [1].

Finally, in the large input field, type the RHS of the truncated normal form map as

```
ksi1'=-ksi1+ksi2
ksl2'=beta1*ksi1-ksi2+beta2*ksi2+CC*ksi1^3+DD*ksi1^2*ksi2
```

Avoid typical mistakes:

- Make sure the multiplication is written explicitly with $*$.
- Specify the right hand sides in the same order as the coordinates.

It is best not to add comma's or semicolons after the equations. Now the **System** window should look like in Figure 2, and you can press the **OK** button. Two new files will be created in the `Systems` directory of MATCONTM, namely the mapfile `Tnfmap.m` and a `mat`-file `Tnfmap.mat`.

[1] If the MATLAB Symbolic Toolbox is present, there will be buttons indicated 'symbolically'. The first-order derivatives are used in some of the integration algorithms, the first- and second-order derivatives are used in the continuation, while the third-order derivatives are employed in the normal form computations. The derivatives of fourth and fifth order are only used in the normal form computations of some codimension 2 bifurcations.

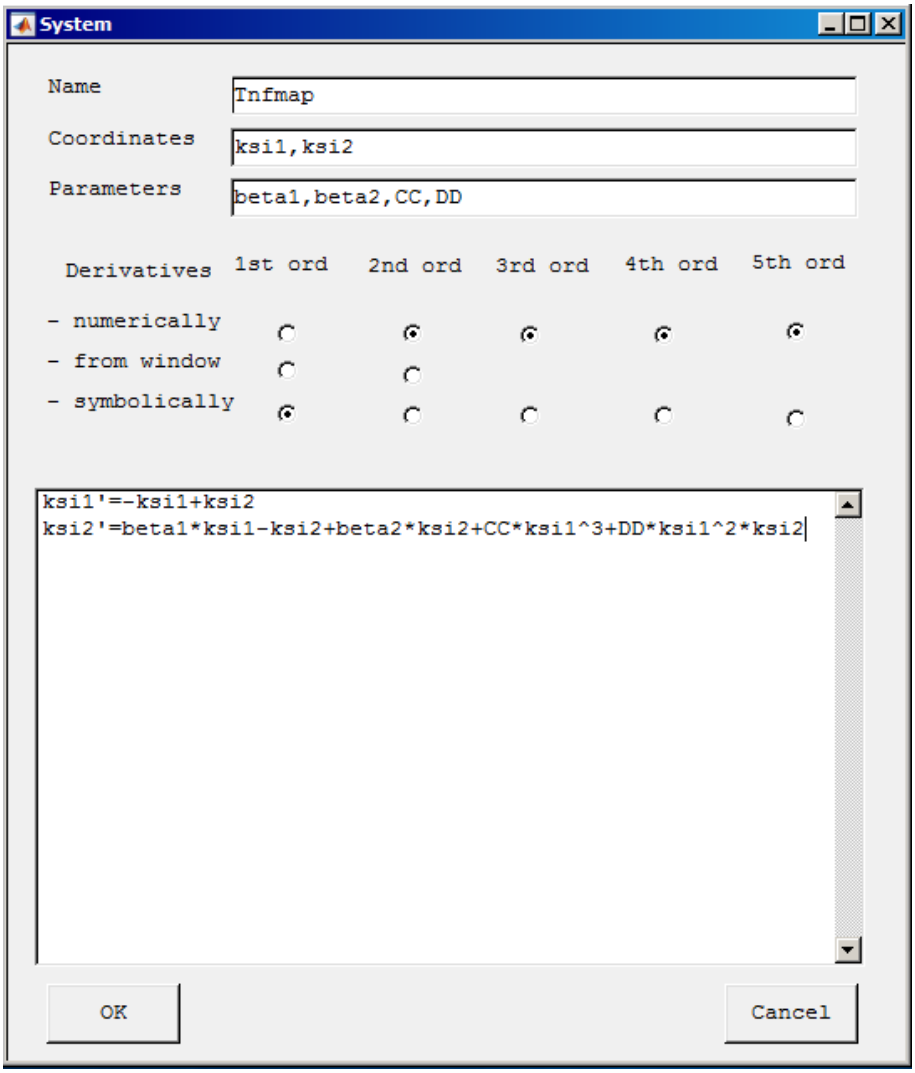


Figure 2: Specifying a new model.

The mapfile can be edited later on by calling `SysGUI.edit(@name)` where `name` is the name of an existing *mapfile*. User functions can be added by calling `SysGUI.userfunctions(@name)`. See Figure 3.

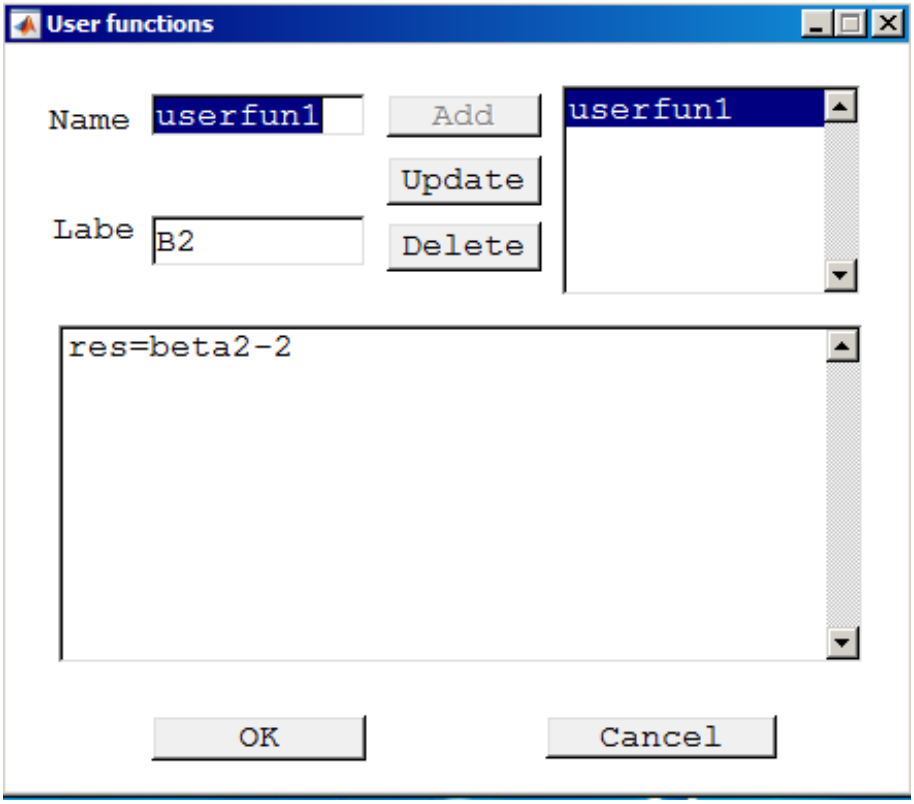


Figure 3: Adding a user function.

First we give the mapfile of M_{TN} using symbolic derivatives up to order 5:

```
function out = Tnfmap
out{1} = [];
out{2} = @fun_eval;
out{3} = @jacobian;
out{4} = @jacobianp;
out{5} = @hessians;
out{6} = @hessiansp;
out{7} = @der3;
out{8} = @der4;
out{9} = @der5;

% --------------------------------------------------------------------
function dydt = fun_eval(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
dydt=[-kmrgd(1)+kmrgd(2);
par_beta1*kmrgd(1)-kmrgd(2)+par_beta2*kmrgd(2)+par_CC*kmrgd(1)^3+par_DD*kmrgd(1)^2*kmrgd(2
```

```
% ------------------------------------------------------------------------
function jac = jacobian(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
jac=[ -1 , 1 ; par_beta1 + 3*kmrgd(1)^2*par_CC + 2*kmrgd(1)*kmrgd(2)*par_DD , par_beta2 +
% ------------------------------------------------------------------------
function jacp = jacobianp(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
jacp=[ 0 , 0 , 0 , 0 ; kmrgd(1) , kmrgd(2) , kmrgd(1)^3 , kmrgd(1)^2*kmrgd(2) ];
% ------------------------------------------------------------------------
function hess = hessians(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
hess1=[ 0 , 0 ; 6*kmrgd(1)*par_CC + 2*kmrgd(2)*par_DD , 2*kmrgd(1)*par_DD ];
hess2=[ 0 , 0 ; 2*kmrgd(1)*par_DD , 0 ];
hess(:,:,1) =hess1;
hess(:,:,2) =hess2;
% ------------------------------------------------------------------------
function hessp = hessiansp(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
hessp1=[ 0 , 0 ; 1 , 0 ];
hessp2=[ 0 , 0 ; 0 , 1 ];
hessp3=[ 0 , 0 ; 3*kmrgd(1)^2 , 0 ];
hessp4=[ 0 , 0 ; 2*kmrgd(1)*kmrgd(2) , kmrgd(1)^2 ];
hessp(:,:,1) =hessp1;
hessp(:,:,2) =hessp2;
hessp(:,:,3) =hessp3;
hessp(:,:,4) =hessp4;
%------------------------------------------------------------------------
function tens3  = der3(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
tens31=[ 0 , 0 ; 6*par_CC , 2*par_DD ];
tens32=[ 0 , 0 ; 2*par_DD , 0 ];
tens33=[ 0 , 0 ; 2*par_DD , 0 ];
tens34=[ 0 , 0 ; 0 , 0 ];
tens3(:,:,1,1) =tens31;
tens3(:,:,1,2) =tens32;
tens3(:,:,2,1) =tens33;
tens3(:,:,2,2) =tens34;
%------------------------------------------------------------------------
function tens4  = der4(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
tens41=[ 0 , 0 ; 0 , 0 ];
tens42=[ 0 , 0 ; 0 , 0 ];
...
tens47=[ 0 , 0 ; 0 , 0 ];
tens48=[ 0 , 0 ; 0 , 0 ];
tens4(:,:,1,1,1) =tens41;
tens4(:,:,1,1,2) =tens42;
...
tens4(:,:,2,2,1) =tens47;
tens4(:,:,2,2,2) =tens48;
%------------------------------------------------------------------------
function tens5  = der5(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
tens51=[ 0 , 0 ; 0 , 0 ];
```

```
tens52=[ 0 , 0 ; 0 , 0 ];
...
tens515=[ 0 , 0 ; 0 , 0 ];
tens516=[ 0 , 0 ; 0 , 0 ];
tens5(:,:,1,1,1,1) =tens51;
tens5(:,:,1,1,1,2) =tens52;
...
tens5(:,:,2,2,2,1) =tens515;
tens5(:,:,2,2,2,2) =tens516;
```

We observe that:

- `out{1}` is empty. This output field is provided for cases in which the user wants to do some initializations.
- The state variables are collected in a vector `kmrgd`.
- Internally the parameters are renamed to avoid clashes with the name restrictions of the symbolic toolbox. E.g., `CC` is replaced by `par_CC`.

After adding a user function userfun1=beta2-2 we obtain:

```
function out = Tnfmap
out{1} = @init;
out{2} = @fun_eval;
out{3} = @jacobian;
out{4} = @jacobianp;
out{5} = @hessians;
out{6} = @hessiansp;
out{7} = @der3;
out{8} = @der4;
out{9} = @der5;
out{10}= @userfun1;

% --------------------------------------------------------------------
function dydt = fun_eval(t,kmrgd,beta1,beta2,CC,DD)
dydt=[-kmrgd(1)+kmrgd(2);
beta1*kmrgd(1)-kmrgd(2)+beta2*kmrgd(2)+CC*kmrgd(1)^3+DD*kmrgd(1)^2*kmrgd(2);];

% --------------------------------------------------------------------
function jac = jacobian(t,kmrgd,beta1,beta2,CC,DD)
jac=[ -1 , 1 ; beta1 + 3*CC*kmrgd(1)^2 + 2*DD*kmrgd(1)*kmrgd(2) , beta2 + DD*kmrgd(1)^2 -
% --------------------------------------------------------------------
function jacp = jacobianp(t,kmrgd,beta1,beta2,CC,DD)
jacp=[ 0 , 0 , 0 , 0 ; kmrgd(1) , kmrgd(2) , kmrgd(1)^3 , kmrgd(1)^2*kmrgd(2) ];
% --------------------------------------------------------------------
function hess = hessians(t,kmrgd,beta1,beta2,CC,DD)
hess1=[ 0 , 0 ; 6*CC*kmrgd(1) + 2*DD*kmrgd(2) , 2*DD*kmrgd(1) ];
hess2=[ 0 , 0 ; 2*DD*kmrgd(1) , 0 ];
```

```
hess(:,:,1) =hess1;
hess(:,:,2) =hess2;
% ------------------------------------------------------------------------
function hessp = hessiansp(t,kmrgd,beta1,beta2,CC,DD)
hessp1=[ 0 , 0 ; 1 , 0 ];
hessp2=[ 0 , 0 ; 0 , 1 ];
hessp3=[ 0 , 0 ; 3*kmrgd(1)^2 , 0 ];
hessp4=[ 0 , 0 ; 2*kmrgd(1)*kmrgd(2) , kmrgd(1)^2 ];
hessp(:,:,1) =hessp1;
hessp(:,:,2) =hessp2;
hessp(:,:,3) =hessp3;
hessp(:,:,4) =hessp4;
%------------------------------------------------------------------------
function tens3  = der3(t,kmrgd,beta1,beta2,CC,DD)
tens31=[ 0 , 0 ; 6*CC , 2*DD ];
tens32=[ 0 , 0 ; 2*DD , 0 ];
tens33=[ 0 , 0 ; 2*DD , 0 ];
tens34=[ 0 , 0 ; 0 , 0 ];
tens3(:,:,1,1) =tens31;
tens3(:,:,1,2) =tens32;
tens3(:,:,2,1) =tens33;
tens3(:,:,2,2) =tens34;
%------------------------------------------------------------------------
function tens4  = der4(t,kmrgd,beta1,beta2,CC,DD)
tens41=[ 0 , 0 ; 0 , 0 ];
tens42=[ 0 , 0 ; 0 , 0 ];
...
tens47=[ 0 , 0 ; 0 , 0 ];
tens48=[ 0 , 0 ; 0 , 0 ];
tens4(:,:,1,1,1) =tens41;
tens4(:,:,1,1,2) =tens42;
...
tens4(:,:,2,2,1) =tens47;
tens4(:,:,2,2,2) =tens48;
%------------------------------------------------------------------------
function tens5  = der5(t,kmrgd,beta1,beta2,CC,DD)
tens51=[ 0 , 0 ; 0 , 0 ];
tens52=[ 0 , 0 ; 0 , 0 ];
...
tens515=[ 0 , 0 ; 0 , 0 ];
tens516=[ 0 , 0 ; 0 , 0 ];
tens5(:,:,1,1,1,1) =tens51;
tens5(:,:,1,1,1,2) =tens52;
...
tens5(:,:,2,2,2,1) =tens515;
tens5(:,:,2,2,2,2) =tens516;
function userfun1=userfun1(t,kmrgd,beta1,beta2,CC,DD)
```

```
userfun1=beta2-2;
```

If no symbolic derivatives are available then MatContM uses finite difference approximations instead. However, this will be less accurate and computed normal form coefficients are often unreliable. A mapfile of M_{TN} without symbolic derivatives is given by:

```
function out = Tnfmap
out{1} = [];
out{2} = @fun_eval;
out{3} = [];
out{4} = [];
out{5} = [];
out{6} = [];
out{7} = [];
out{8} = [];
out{9} = [];
out{10}= @userfun1;

% --------------------------------------------------------------------
function dydt = fun_eval(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
dydt=[-kmrgd(1)+kmrgd(2);
par_beta1*kmrgd(1)-kmrgd(2)+par_beta2*kmrgd(2)+par_CC*kmrgd(1)^3+par_DD*kmrgd(1)^2*kmrgd(2

% --------------------------------------------------------------------
function jac = jacobian(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
% --------------------------------------------------------------------
function jacp = jacobianp(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
% --------------------------------------------------------------------
function hess = hessians(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
% --------------------------------------------------------------------
function hessp = hessiansp(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
%--------------------------------------------------------------------
function tens3  = der3(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
%--------------------------------------------------------------------
function tens4  = der4(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
%--------------------------------------------------------------------
function tens5  = der5(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
function userfun1=userfun1(t,kmrgd,par_beta1,par_beta2,par_CC,par_DD)
userfun1=beta2-2
```

A special point on a bifurcation curve that is specified by a user function has a structure as follows:

s.index	index of the detected singular point defined by the user function.
s.label	a string that is in `UserInfo.label`, label of the singularity.
s.data	an empty tangent vector, values of the test and user functions in the singular point.
s.msg	a string that is set in `UserInfo.name`.

2.7 Computing orbits and the Jacobian

The main use of the mapfile is in the continuation and bifurcation algorithms but it can also be used to apply the map, compute orbits and Jacobians. Computing long orbits is an important feature since it often allows to find stable fixed points and stable cycles from scratch.

The most direct way to apply the map is shown in the testrun `Tnfmap0.m` in the directory `Testruns/Tnfmap`:

```
funchandle=feval(@Tnfmap);
func=funchandle{2};
x=[0.4;0.6];
p=[1;2;3;4];
P0=num2cell(p);
func(0,x,P0{:})
```

By running this script one computes $f(x, p)$ where f is the map that is introduced in the mapfile `Tnfmap.m`. The output is

```
>> Tnfmap0

ans =

    0.2000
    1.5760
```

A less straightforward but more powerful way is to use the object `MatContMSystem.m` in the directory `Systems`. An example is provided in the testrun `Tnfmap00` in the directory `Testruns/Tnfmap`:

```
f=MatContMSystem(@Tnfmap)
f([0.4;0.6],[1 2 3 4])
f.orbit(5,[0.4;0.6],[1 2 3 4])
f.jacobian([0.4;0.6],[1 2 3 4])
f4=f.getIteratedMap(4)
f4([0.4;0.6],[1 2 3 4])
```

Running this file applies the map, computes an orbit of length 5, computes the Jacobian, defines the fourth-iterate map and applies it. The output is as follows:

```
>> Tnfmap00

f =

  MatContMSystem with properties:

    iteration: 1
```

```
ans =

    0.2000
    1.5760


ans =

    0.4000    0.2000    1.3760    0.6762   26.1099
    0.6000    1.5760    2.0522   26.7861   77.3751


ans =

   -1.0000    1.0000
    4.3600    1.6400


f4 =

  MatContMSystem with properties:

    iteration: 4


ans =

   26.1099
   77.3751
```

The Jacobian of an iterate of the map can be computed in the same way.

3 Continuation of cycles

The subdirectory `FixedPointMap` contains all files related to the continuation of fixed points of iterates of maps, detection of their bifurcations, computation of the normal forms etc. The main files are `fixedpointmap.m`, `init_FPm_FPm.m` and `init_BPm_FPm.m` . The data specific to this continuation are stored in the global structure *fpmds*. Such a continuation can be found, for instance, in §10.1, *Run 1*.

3.1 Bifurcations and test functions

Consider

$$f^i : x \longrightarrow f^i(x) = f(f(..f(x))) \tag{5}$$

The iteration gives rise to a sequence of points

$$x = x_1, x_2, x_3, ..x_{i+1},$$

in which $x_{l+1} = f^l(x_1)$. The equation for fixed points of the i-th iterate is

$$f^i(x) - x = 0 \tag{6}$$

There are three generic codim 1 bifurcations that can be detected along the curve of fixed points of the i-th iterate, namely Fold (Limitpoint, `LP`), Flip (Period Doubling, `PD`) and Neimark-Sacker (`NS`). Also, there can be branch points (`BP`). We consider them in this order. To detect these singularities, we define 4 test functions.

$$\phi_1(x, \alpha) = \det(A^{(i)} \odot A^{(i)} - I_m), \tag{7}$$
$$\phi_2(x, \alpha) = \det(A^{(i)} + I_n), \tag{8}$$
$$\phi_3(x, \alpha) = v_{n+1}, \tag{9}$$
$$\phi_4(x, \alpha) = \det \begin{pmatrix} F_X \\ v^T \end{pmatrix}. \tag{10}$$

Here v is the tangent vector along the curve, $\odot$ is the $m \times m$ bialternate matrix product where $m = \frac{n(n-1)}{2}$ (cf. [22], §4.4.4), $F(X) = f^i(x, \alpha) - x$ and $A^{(i)}$ is the Jacobian matrix of the iterated map f^i.

The following codimension 1 bifurcations and branch points can be detected and located as regular zeroes of the above test functions:

- NS: $\phi_1 = 0$.
- PD: $\phi_2 = 0$.
- LP: $\phi_3 = 0, \phi_4 \neq 0$.
- BP: $\phi_4 = 0$.

We notice that ϕ_1 is also zero if there is a pair of real multipliers with product 1. Such points are called neutral saddles. We have to take care of these when processing the NS points.
The singularity matrix is :

$$S = \begin{pmatrix} 0 & - & - & - \\ - & 0 & - & - \\ - & - & 0 & 1 \\ - & - & - & 0 \end{pmatrix} \tag{11}$$

3.2 Fixed point initializations

The initializers are `init_FPm_FPm.m`, `init_PDm_FP2m.m` and `init_BPm_FPm.m`. To start from a known fixed point of the i-th iterate one first gives the following curve initializer statement:

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_FPm\_FPm}(@\texttt{mapfile}, \texttt{x}, \texttt{p}, \texttt{ap}, \texttt{i})$$

Here `mapfile` is the mapfile to be used, `x` is a vector containing the starting values of the state variables, `p` is the vector containing the starting values of the parameters and `ap` is the index of the active parameter. This routine stores its output partly in the global structure `fpmds`. The output of `init_FPm_FPm` contains a vector x_0 with the state variables and the active parameter and an empty vector v_0.

To explain the meaning of *fpmds* we run the fixed point initializer using the *mapfile* that is defined in §2.6 where symbolic derivatives are used. The global structure *fpmds* is set using:

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_FPm\_FPm}(@\texttt{Tnfmap}, [0; 0], [-1; 0; 1; 1], 2, 1)$$

Some important fildes of *fpmds* are given by:

```
           P0: [4x1 double]
 ActiveParams: 2
      mapfile: @Tnfmap
         func: @fun_eval
     Jacobian: @jacobian
    JacobianP: @jacobianp
     Hessians: @hessians
    HessiansP: @hessiansp
         Der3: @der3
         Der4: @der4
         Der5: @der5
  Niterations: 1
       nphase: 2
```

To start the continuation of 2-cycles from a period-doubling point detected during a fixed point of i-th iterate continuation one first gives the following curve initializer statement:

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_PDm\_FP2m}(@\texttt{mapfile}, \texttt{xnew}, \texttt{p}, \texttt{ap}, \texttt{s(j)}, \texttt{h}, \texttt{i})$$

Here `mapfile` is the mapfile to be used, `xnew` is a vector containing the starting values of the state variables, `p` is the vector containing the starting values of the parameters and `ap` must be the index of the active parameter. In the most natural situation where `x` is the matrix returned by the previous fixed point curve continuation one starts to build `xnew` by the statement `xnew=x(1:nphase,s(j).index)`.
`s(j)` is the special point structure of the detected period doubling point on the fixed point of i-th iterate curve continuation and `nphase` is the number of state variables.
Next, the statement

```
p(ap_old)=x(end,s(j).index);
```

replaces the old value of the free parameter in the previous run by the parameter value at the `PD` point.
The output of `init_PDm_FP2m` contains a vector x_0 with the state variables and the active parameter and a tangent vector v_0. If x_{PD} is the `PD` point on the original branch and q is the right eigenvector of the multiplier -1 in x_{PD} then $x_0 = x_{PD} + hq$ and $v_0 = q$; the scalar h is called the amplitude.

We note that `init_PDm_FP2m` uses the global structures `fpmds` and `cds` that were created during the computation of the fixed point curve on which the `PD` point was detected. This should be taken into account if intermediate computations are performed. The routine `init_PDm_FP2m` stores its output partly in the same global structures `fpmds` and `cds`.

3.3 Output of a fixed point continuation

The fixed point curve is continued by calling:

$$[\mathtt{x}, \mathtt{v}, \mathtt{s}, \mathtt{h}, \mathtt{f}] = \mathtt{cont}(@\mathtt{fixedpointmap}, \mathtt{x0}, \mathtt{v0}, \mathtt{opt}) \tag{12}$$

This call returns :
`x` and `v`: points and their tangent vectors along the fixed points curve, respectively.
The array `s` contains information about the computed singular points, including zeros of user functions, with the following fields:

s.index	index of the point in `x`.
s.label	label of the singularity, may be `00`, `NS`, `PD`, `LP`, `BP`, `99` or the label of a user function. The strings `00` and `99` indicate the first and the last point on the fixed points curve, respectively.
s.data	extra information. For the first and last points this is only an empty tangent vector. For zeroes of user functions an empty tangent vector is given, plus the values of all active user functions and test functions. For bifurcation points see the respective cases in §3.4.1, §3.4.2, §3.4.3, §3.5.
s.msg	a string containing a message for this particular singularity. For the first and last points these are the strings 'This is the first point of the curve' and 'This is the last point of the curve', respectively. For zeroes of user functions the name of the user function is given. For bifurcation points see again the respective cases.

`h` and `f` were described in §2.5.1

3.4 Normal form coefficients of codim-1 bifurcation points

When a limit point, period doubling point or Neimark-Sacker point is detected on a curve of fixed points, then the processing of these points includes the computation of the normal form coefficients.
Assuming sufficient smoothness of f, we write

$$\begin{aligned} f^r(x_0+u,\alpha_0) \;&=\; x_0 + A^{(r)}u + \tfrac{1}{2}\,B^{(r)}(u,u) + \tfrac{1}{6}\,C^{(r)}(u,u,u) \\ &+\; \tfrac{1}{24}\,D^{(r)}(u,u,u,u) + \tfrac{1}{120}\,E^{(r)}(u,u,u,u,u) + O(\|u\|^6), \end{aligned} \tag{13}$$

where $A^{(r)} = (f^r)_x(x_0)$ and the components of the multilinear functions $B^{(r)}, C^{(r)}, D^{(r)}$, and $E^{(r)}$ are given by

$$\begin{aligned}
B^{(r)}(x,y) &= \sum_{j,k=1}^{n} \frac{\partial^2 f^{(r)}(x_0,\alpha_0)}{\partial\xi_j\partial\xi_k} x_j y_k, \\
C^{(r)}(x,y,z) &= \sum_{j,k,l=1}^{n} \frac{\partial^3 f^{(r)}(x_0,\alpha_0)}{\partial\xi_j\partial\xi_k\partial\xi_l} x_j y_k z_l, \\
D^{(r)}(x,y,z,u) &= \sum_{j,k,l,m=1}^{n} \frac{\partial^4 f^{(r)}(x_0,\alpha_0)}{\partial\xi_j\partial\xi_k\partial\xi_l\partial\xi_m} x_j y_k z_l u_m, \\
E^{(r)}(x,y,z,u,v) &= \sum_{j,k,l,m,s=1}^{n} \frac{\partial^5 f^{(r)}(x_0,\alpha_0)}{\partial\xi_j\partial\xi_k\partial\xi_l\partial\xi_m\partial\xi_s} x_j y_k z_l u_m v_s,
\end{aligned}$$

for $r = 1, 2, \ldots, n$. Here $||x|| = \sqrt{\langle x, x\rangle} = \sqrt{x^T x}$. We now give the critical normal form coefficients of the generic codimension 1 bifurcations of fixed points. For details and proofs we refer to [33], §5.4.2.

3.4.1 Limit point

The matrix $A^{(i)}$ (Jacobian of the i-th iterate) has a simple eigenvalue $\lambda_1 = 1$ and no other multipliers on the unit circle, while the restriction of (1) to a one dimensional center manifold at the critical parameter value has the form

$$w \longmapsto w + aw^2 + O(w^3), w \in \mathbb{R}^1 \tag{14}$$

if $a \neq 0$, where for the coefficient a we have the expression:

$$a = \frac{1}{2}\left\langle p, B^{(i)}(q,q)\right\rangle, \tag{15}$$

where $A^{(i)}q = q, (A^{(i)})^T p = p$, and $\langle q, q\rangle = 1$, $\langle p, q\rangle = 1$.
A generic unfolding of (14) is

$$w \longmapsto \alpha + w + aw^2 + O(w^3), w \in \mathbb{R}^1 \tag{16}$$

where α is the control parameter with critical value 0. When the control parameter crosses 0, two fixed points collide and disappear. So the fixed point curve has a turning point with respect to the control parameter.
At a fold point the output `s`, in the continuation of fixed point curve (12), contains:

s.index	index of the fold point.
s.label	the label `LP`.
s.data	an empty tangent vector, values of the active test functions and active user functions, the normal form coefficient a, multipliers and corresponding eigenvectors.
s.msg	the string `Limit point`.

3.4.2 Period doubling

The matrix $A^{(i)}$ has a simple eigenvalue $\lambda_1 = -1$ and no other multipliers on the unit circle. The restriction of (1) to a one dimensional center manifold at the critical parameter value can be transformed to the normal form

$$w \longmapsto -w + bw^3 + O(w^4), w \in R^1 \tag{17}$$

if $b \neq 0$, where b is given by

$$b = \frac{1}{6}\left\langle p, C^{(i)}(q,q,q) + 3B^{(i)}(q,(I-A)^{-1}B^{(i)}(q,q))\right\rangle, \tag{18}$$

where I is the unit $n \times n$ matrix, $A^{(i)}q = -q, (A^{(i)})^T p = -p$, $\langle q,q\rangle = 1$,$\langle p,q\rangle = 1$.
A generic unfolding of (17) is

$$w \longmapsto -w(1+\alpha) + bw^3 + O(w^4), w \in R^1 \tag{19}$$

where α is a control parameter. When the control parameter crosses the critical value 0, a cycle of period 2 bifurcates from the fixed point. If $b > 0$ then this period two cycle is stable and found for those α - values where the fixed point of the map is unstable; this is called a supercritical `PD`. If $b < 0$ then the period two cycle is unstable and found for those α - values where the fixed point of the map is stable; this is called a subcritical `PD` point. At a flip point the output `s`, in the continuation of fixed point curve (12), contains:

s.index	index of the flip point.
s.label	the label `PD`.
s.data	an empty tangent vector, values of the active test functions, values of the active user functions, the right eigenvector, the normal form coefficient b, multipliers and corresponding eigenvectors.
s.msg	the string `Period doubling`.

3.4.3 Neimark-Sacker

The matrix $A^{(i)}$ has simple critical multipliers $\lambda_{1,2} = e^{\pm i\theta_0}$ $(0 \leq \theta \leq \pi)$ and no other multipliers on the unit circle. Assume that $e^{ik\theta_0} \neq 1, k = 1,2,3,4$ (these special cases are the *strong resonances*, §5.4-7).

Then the restriction of (1) to the two dimensional center manifold at the critical parameter value can be transformed to the normal form

$$w \longmapsto we^{i\theta_0}(1 + d|w|^2) + O(|w|^4), w \in C^1$$

where w is now a complex variable and d is a complex number. If $c = Re(d) \neq 0$, then a unique closed invariant curve around the fixed point appears when the parameter crosses the critical value. One has the following expression for d:

$$d = \frac{1}{2}e^{-i\theta_0}\left\langle p, C^{(i)}(q,q,\bar{q}) + 2B^{(i)}(q,h_{11}) + B^{(i)}(\bar{q},h_{20})\right\rangle \tag{20}$$

where

$$h_{11} = (I_n - A)^{-1}B^{(i)}(q,\bar{q}), \quad h_{20} = (e^{2i\theta_0}I_n - A^{(i)})^{-1}B^{(i)}(q,q),$$

and $A^{(i)}q = e^{i\theta_0}q,\ [A^{(i)}]^T p = e^{-i\theta_0}p$ and $\langle q, q\rangle = \langle p, q\rangle = 1$.
If $c < 0$ then an stable invariant curve branches off the `NS` point and is found for values of the control parameter for which the fixed point of the map is unstable.

If $c > 0$ then an unstable invariant curve branches off the `NS` point and is found for values of the control parameter for which the fixed point of the map is stable. At a Neimark-Sacker point the output `s`, in the continuation of fixed point curve (12), contains:

s.index	index of the Neimark-Sacker point.
s.label	the label `NS`.
s.data	an empty tangent vector, values of the active test functions and user functions, the normal form coefficient c, multipliers and corresponding eigenvectors. a field ' mprocess_NS' that is empty or 'Neutral_ saddle' in the case of Neimark-Sacker or Neutral-Saddle point, respectively.
s.msg	either the string `Neimark_Sacker` or `Neutral saddle`.

3.5 Branch switching

In this section we consider the approximation of a new cycle curve that emanates from a branch point for (6). This situation is very similar to that of branch points of equilibria. We consider the iterate map $f^i : \mathbb{R}^{n+1} \longmapsto \mathbb{R}^n$. A solution $X_0 = X(s_0)$ of

$$F(x) = f^i(x) - x = 0 \tag{21}$$

is called a simple singular point if $F_X(X_0)$ has rank $n-1$.
At a branch point point the output `s`, in the continuation of fixed point curve (12), contains:

s.index	index of the branch point.
s.label	the label `BP`.
s.data	tangent vector, values of the active test functions and user functions, multipliers and corresponding eigenvectors.
s.msg	the string `Branch point`.

To initialize the continuation of the new branch in a branch point a special routine is needed. Its syntax is

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_BPm\_FPm}(@\texttt{mapfile}, \texttt{x}, \texttt{p}, \texttt{s(j)}, \texttt{h}, \texttt{i})$$

This routine calculates an initial point for starting a new branch from a branch point detected on a fixed point curve. Here *mapfile* is the mapfile to be used, x is an array containing the values of the state variables returned by the previous fixed point curve continuation. p is the vector containing the current values of the parameters, $s(j)$ is the special point structure where j is the index of the branch point, h contains the value of the initial amplitude and i is the iteration number. In most cases, i is the iteration number of the branch of fixed points on which the `BP` point x_{BP} was detected. If q is the unit vector in the direction of the new branch in x_{BP} then $x_0 = x_{BP} + hq$ and $v_0 = q$; the scalar h is called the amplitude.

We notice that `init_BPm_FPm` uses the global structures `fpmds` and `cds` created during the continuation of the fixed point curve on which the `BP` was detected. This should be taken into account if intermediate computations are performed. This routine stores part of its output

in the global structures `fpmds` and `cds`. Such a branch switching is presented, for instance, in §10.1, *Run 2*.

3.6 Fixed point curve adaptation

In the continuation of fixed points there is no need to change the system, so adaptation is void in this case.

4 Continuation of codim-1 bifurcations of cycles

4.1 Continuation of fold curves

The files which are specific to this type of computation are stored in the directory `LimitPointMap`. The main files in this directory are `limitpointmap.m` and `init_LPm_LPm.m`. The data specific to this continuation are stored in the global structure *lpmds*. Such a continuation can be found, for instance, in §10.2, *Run 18*.

4.1.1 The defining system

If $A^{(i)}(x,\alpha)$ is the Jacobian matrix of f^i evaluted in (x,α), then we have the continuation problem

$$\left\{\begin{array}{rcl} f^i(x,\alpha)-x & = & 0,\\ det(A^{(i)}(x,\alpha)-I_n) & = & 0,\end{array}\right. \tag{22}$$

which is a system of $n+1$ equations in an $(n+2)$ -dimensional space with coordinates (x,α). In MatContM limit point curves are computed by *minimally extended defining systems*, cf. [22], §4.1.2. The limit point curve is defined by the following system

$$\left\{\begin{array}{rcl} f^i(x,\alpha)-x & = & 0,\\ g(x,\alpha) & = & 0,\end{array}\right. \tag{23}$$

where $(x,\alpha)\in\mathbf{R}^{n+2}$, while g is obtained by solving

$$\left(\begin{array}{cc} f^i_x(x,\alpha)-I_n & w_{bor}\\ v_{bor}^T & 0\end{array}\right)\left(\begin{array}{c} v\\ g\end{array}\right)=\left(\begin{array}{c} 0_n\\ 1\end{array}\right), \tag{24}$$

and $w_{bor}, v_{bor}\in\mathbb{R}^n$ are chosen such that the matrix in (24) is nonsingular. An advantage of this method is that the derivatives of g can be obtained easily from the derivatives of $f^i_x(x,\alpha)$:

$$g_z=-w^T(f^i_x)_z v \tag{25}$$

where z is a state variable or an active parameter and w is obtained by solving

$$\left(\begin{array}{cc} (f^i_x)^T(x,\alpha)-I_n & v_{bor}\\ w_{bor}^T & 0\end{array}\right)\left(\begin{array}{c} w\\ g\end{array}\right)=\left(\begin{array}{c} 0_n\\ 1\end{array}\right). \tag{26}$$

We note that the quantities called g in (24) and (26) are the same since they are both equal to the bottom right element of the inverse of the square matrix in (24). This method is implemented in the curve definition file `limitpointmap.m`.

4.1.2 Bifurcations and test functions

There are four generic codim 2 bifurcations that can be detected along the limit point curve:

- *Resonance 1:1.* We will denote this bifurcation with `R1`
- *Fold+Flip* point, denoted as `LPPD`
- *Fold+Neimark-Sacker* point, denoted as `LPNS`
- *Cusp* point, denoted as `CP`

To detect these singularities, we first define 4 test functions:

- $\phi_1 = w^T v$
- $\phi_2 = det(A^{(i)}(x, \alpha) + I_n)$
- $\phi_3 = \det(A^{(i)} \odot A^{(i)} - I_m)$
- $\phi_4 = \langle w, B^{(i)}(v, v) \rangle$

In these expressions v and w are the vectors computed in (24) and (26), respectively. The singularity matrix is:

$$S = \begin{pmatrix} 0 & - & 0 & - \\ - & 0 & - & - \\ 1 & - & 0 & - \\ - & - & - & 0 \end{pmatrix} \tag{27}$$

4.1.3 Fold initialization

The only way to start a fold curve continuation in the toolbox is from a limit point. To initialize the continuation one first gives the following statement:

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_LPm\_LPm}(@\texttt{mapfile}, \texttt{xnew}, \texttt{p}, \texttt{ap}, \texttt{i})$$

In this statement xnew must be a vector that contains the values of the state variables. p must contain the current values of all the parameters and ap must be the indices of the 2 active parameters. In the most natural situation where x is the matrix returned by the previous equilibrium curve continuation one starts to build x_{new} by the statement
$xnew = x(1 : nphase, s(j).index)$. $s(j)$ is the special point structure of the detected fold point on the equilibrium curve continuation and nphase is the number of state variables. Now x_{new} contains the state variables. Next, the statement
`p(ap_old)=x(end,s(j).index)`
replaces the value of the free parameter in the previous run by its value at the fold point. *mapfile* specifies the mapfile to be used. The output of `init_LPm_LPm` contains a vector x_0 with the state variables and the active parameters and an empty vector v_0.

4.1.4 Output of a fold continuation

The fold curve is continued by calling:

$$[\texttt{x}, \texttt{v}, \texttt{s}, \texttt{h}, \texttt{f}] = \texttt{cont}(@\texttt{limitpointmap}, \texttt{x0}, \texttt{v0}, \texttt{opt}) \tag{28}$$

This call returns :
`x` and `v`: points and their tangent vectors along the fold curve, respectively.
The array `s` contains information about the computed singularity points with the following fields:

s.index	index of the singularity point in x.
s.label	label of the singularity, may be `R1`, `LPPD`, `LPNS` and `CP`.
s.data	values of the active test functions and user functions, the normal form coefficients, see the respective cases in §5.4, §5.8, §5.9, §5.1, multipliers and corresponding eigenvectors.
s.msg	strings can be *Resonance 1:1*, *Fold+Flip*, *Fold+Neimark-Sacker* and *Cusp*.

`h` and `f` were discussed in §2.5.1.

4.1.5 Adaptation

It is possible to adapt the problem while generating the fold curve. This call updates the auxiliary variables used in the defining system of the computed branch. The bordering vectors v_{bor} and w_{bor} may require updating since they must at least be such that the matrices in (24), (26) are nonsingular. Updating is done by replacing v_{bor} and w_{bor} by the normalized vectors v, w computed in (24), (26), respectively.

4.2 Continuation of flip curves

The files which are specific to this type of computation are stored in the directory `PeriodDoublingMap`. The main files in this directory are `perioddoublingmap.m` and `init_PDm_PDm.m`. The data specific to this continuation are stored in the global structure `pdmds`. Such a continuation can be found, for instance, in §10.2, *Run 16*.

4.2.1 The defining system

If $A^{(i)}(x,\alpha)$ is the Jacobian matrix of f^i in (5) evaluted in (x,α), then we get the continuation problem

$$\left\{ \begin{array}{rcl} f^i(x,\alpha) - x & = & 0, \\ det(A^{(i)}(x,\alpha) + I_n) & = & 0, \end{array} \right. \tag{29}$$

which is a system of $n+1$ equations in an $(n+2)$ -dimensional space with coordinates (x,α). In MatContM period doubling point curves are computed by *minimally extended defining systems* cf. [22], §4.1.2. The limit point curve is defined by the following system

$$\left\{ \begin{array}{rcl} f^i(x,\alpha) - x & = & 0, \\ g(x,\alpha) & = & 0, \end{array} \right. \tag{30}$$

where $(x,\alpha) \in \mathbf{R}^{n+2}$, and g is obtained by solving

$$\left(\begin{array}{cc} f^i_x(x,\alpha) + I_n & w_{bor} \\ v_{bor}^T & 0 \end{array} \right) \left(\begin{array}{c} v \\ g \end{array} \right) = \left(\begin{array}{c} 0_n \\ 1 \end{array} \right), \tag{31}$$

and $w_{bor}, v_{bor} \in \mathbb{R}^n$ are chosen such that the matrix in (31) is nonsingular.

An advantage of this method is that the derivatives of g can be obtained easily from the derivatives of $f_x^i(x,\alpha)$:

$$g_z = -w^T (f_x^i)_z v \tag{32}$$

where z is a state variable or an active parameter and w is obtained by solving

$$\begin{pmatrix} (f_x^i)^T(x,\alpha) + I_n & v_{bor} \\ w_{bor}^T & 0 \end{pmatrix} \begin{pmatrix} w \\ g \end{pmatrix} = \begin{pmatrix} 0_n \\ 1 \end{pmatrix}, \tag{33}$$

This method is implemented in the curve definition file `perioddoubling.m`.

4.2.2 Bifurcations and test functions

In discrete maps there are four generic codim 2 bifurcations that can be detected along the period doubling curve:

- *Resonance 1:2* point, denoted as `R2`
- *Fold+Flip* point, denoted as `LPPD`
- *Flip+Neimark-Sacker* point, denoted as `PDNS`
- *Generalized flip* point, denoted as `GPD`

To detect these singularities, we define 4 test functions:

- $\phi_1 = w^T v$
- $\phi_2 = det(A^{(i)}(x,\alpha) - I_n)$
- $\phi_3 = \det(A^{(i)} \odot A^{(i)} - I_m)$
- $\phi_4 =< w, C^{(i)}(v,v,v) > +3 < w, B^{(i)}(v, (I_n - A^{(i)})^{-1} B^{(i)}(v,v)) >$

In these expressions v and w are the vectors computed in (31),(33), respectively.

The singularity matrix for is:

$$S = \begin{pmatrix} 0 & - & 0 & - \\ - & 0 & - & - \\ 1 & - & 0 & - \\ - & - & - & 0 \end{pmatrix} \tag{34}$$

4.2.3 Period doubling initialization

The only way to start a period doubling curve continuation in the toolbox is from a period doubling point. To initialize the continuation one first gives the following statement:

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_PDm\_PDm}(@\texttt{mapfile}, \texttt{xnew}, \texttt{p}, \texttt{ap}, \texttt{i})$$

In this statement `xnew` must be a vector that contains the values of the state variables. `p` must contain the current values of all the parameters and `ap` must be the indices of the active parameters. In the most natural situation where `x` is the matrix returned by the previous

equilibrium curve continuation one starts to build x_{new} by the statement
`xnew=x(1:nphase,s(j).index)`. `s(j)` is the special point structure of the detected period doubling point on the equilibrium curve continuation and `nphase` is the number of state variables. Now x_{new} contains the state variables. Next, the statement
`p(ap_old)=x(end,s(j).index);` replaces the old value of the free parameter in the previous run by the newly found parameter `p`. `mapfile` specifies the ode-file to be used. The output of `init_PDm_PDm` contains a vector x_0 with the state variables and the active parameters and an empty vector v_0.

4.2.4 Output of a flip continuation

The flip curve is continued by calling:

$$[\mathtt{x}, \mathtt{v}, \mathtt{s}, \mathtt{h}, \mathtt{f}] = \mathtt{cont}(@\mathtt{perioddoublingmap}, \mathtt{x0}, \mathtt{v0}, \mathtt{opt})$$

This call returns :
`x` and `v`: points and their tangent vectors along the flip curve, respectively.
The array `s` contains information about the computed singularity points with the following fields:

s.index	index of the singularity point in x.
s.label	label of the singularity, may be `R2`, `LPPD`, `PDNS` and `GPD`.
s.data	values of the active test functions and user functions, the normal form coefficients, see the respective cases in §5.5, §5.8, §5.10, §5.2, multipliers and corresponding eigenvectors.
s.msg	strings can be *Resonance 1:2*, *Fold+Flip*, *Flip+Neimark-Sacker* and *Degenerate Flip*.

`h` and `f` were discussed in §2.5.1.

4.2.5 Adaptation

It is possible to adapt the problem while generating the period doubling curve. This call updates the auxiliary variables used in the defining system of the computed branch. The bordering vectors v_{bor} and w_{bor} may require updating since they must at least be such that the matrices in (31), (33) are nonsingular. Updating is done by replacing v_{bor} and w_{bor} by the normalized vectors v, w computed in (31), (33), respectively.

4.3 Continuation of Neimark-Sacker curves

The files which are specific to this type of computation are stored in the directory `NeimarkSackerMap`. The main files in this directory are `neimarksackermap.m` and `init_NSm_NSm.m`. The data specific to this continuation are stored in the global structure `nsmds`. Such a continuation can be found, for instance, in §10.3, *Run 3*.

4.3.1 The defining system

The continuation variables for a Neimark-Sacker continuation consist of the state and parameter values, and the scalar k defined as the real part of the Neimark-Sacker multipliers $e^{\pm i\theta}$.

The Neimark-Sacker curve is defined by the following system

$$\left\{ \begin{array}{lcl} f^i(x,\alpha) - x & = & 0 \\ g_{i_1 j_1}(x,\alpha,k) & = & 0 \\ g_{i_2 j_2}(x,\alpha,k) & = & 0, \end{array} \right. \tag{35}$$

i.e. by $n+2$ equations in the $(n+3)$ unknowns $x \in R^n$, $\alpha \in R^2$, $k \in R$. Here $(i_1, j_1, i_2, j_2) \in \{1,2\}$ and $g_{i,j}$ are the components of G:

$$G = \begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix}$$

which is obtained by solving

$$\begin{pmatrix} (A^i)^2(x,\alpha) - 2kA^i(x,\alpha) + I_n & W_{bor} \\ V_{bor}^T & O \end{pmatrix} \begin{pmatrix} V \\ G \end{pmatrix} = \begin{pmatrix} 0_{n,2} \\ I_2 \end{pmatrix}, \tag{36}$$

$V_{bor}, W_{bor} \in R^{n \times 2}$ and the integers i_1, j_1, i_2, j_2 are chosen (and can be adapted) so that the matrix in (36) is nonsingular.

This method is implemented in the curve definition file `neimarksackermap.m`.

4.3.2 Bifurcations and test functions

The bifurcations that can be detected along the Neimark-Sacker curve are:

- *Chenciner* point, denoted as `CH`: $\phi_1 = 0$
- *Flip+Neimark-Sacker* point, denoted as `PDNS`: $\phi_2 = 0$; $\phi_6 \neq 0$
- *Fold+Neimark-Sacker* point, denoted as `LPNS`: $\phi_3 = 0$; $\phi_4 \neq 0$
- *Resonance 1:1* point, denoted as `R1` : $\phi_3 = \phi_4 = 0$
- *Double Neimark-Sacker* point, denoted as `NSNS`: $\phi_5 = 0$
- *Resonance 1:2* point, denoted as `R2`: $\phi_2 = \phi_6 = 0$
- *Resonance 1:3* point, denoted as `R3`: $\phi_7 = 0$
- *Resonance 1:4* point, denoted as `R4` : $\phi_8 = 0$

To detect these singularities, we define 8 test functions:

- $\phi_1 = Re(d)$ (see formula (20))
- $\phi_2 = \det(A^{(i)} + I_n)$
- $\phi_3 = \det(A^{(i)} - I_n)$
- $\phi_4 = k - 1$
- $\phi_5 = \det(A^{(i)}|_{N^C} \odot A^{(i)}|_{N^C})$
- $\phi_6 = k + 1$

- $\phi_7 = k + \frac{1}{2}$
- $\phi_8 = k$.

Where $A^{(i)} = (f^i)_x(x, \alpha)$ and the vectors $p, q \in \mathbb{C}^n$ satisfy

$$A^{(i)}q = e^{i\theta},\ (A^{(i)})^T p = e^{-i\theta},\ \langle \text{Re } q, \text{Im } q\rangle = 0,\ \langle q, q\rangle = \langle p, q\rangle = 1$$

The subspace N^C of $\mathbb{R}^n$ is the orthogonal complement of the critical two-dimensional left eigenspace associated with the pair of multipliers with unit product; $2m = (n-2)(n-3)$. In this case the singularity matrix is:

$$S = \begin{pmatrix} 0 & - & - & - & - & - & - & - \\ - & 0 & - & - & - & 1 & - & - \\ - & - & 0 & 1 & - & - & - & - \\ - & - & 0 & 0 & - & - & - & - \\ - & - & - & - & 0 & - & - & - \\ - & 0 & - & - & - & 0 & - & - \\ - & - & - & - & - & - & 0 & - \\ - & - & - & - & - & - & - & 0 \end{pmatrix} \tag{37}$$

4.3.3 Neimark-Sacker initialization

The only way to start the continuation of a Neimark-Sacker bifurcation curve is to start it from a Neimark-Sacker point, typically found on a curve of fixed points. The initializer is called as follows:

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_NSm\_NSm}(@\texttt{mapfile}, \texttt{xnew}, \texttt{p}, \texttt{ap}, \texttt{i})$$

The input arguments `mapfile, xnew, p`, `ap` are built exactly as in `init_LPm_LPm`. The output vector `x0` consists of x_{new} extended with the free parameters and the k in (36).

4.3.4 Output of a Neimark-Sacker continuation

The continuation is then performed by the command

$$[\texttt{x}, \texttt{v}, \texttt{s}, \texttt{h}, \texttt{f}] = \texttt{cont}(@\texttt{neimarksackermap}, \texttt{x0}, \texttt{v0}, \texttt{opt})$$

This call returns :
`x` and `v`: points and their tangent vectors along the Neimark-Sacker curve, respectively.
The array `s` contains information about the computed singular points with the following fields:

s.index	index of the singularity point in x.
s.label	label of the singularity, may be `CH`, `PDNS`, `LPNS`, `R1`, `NSNS`, `R2`, `R3` and `R4`.
s.data	values of the active test functions and user functions, the normal form coefficients, see the respective cases in §5.3, §5.10, §5.9, §5.4, §5.11, §5.5, §5.6, §5.7, multipliers and corresponding eigenvectors.
s.msg	strings can be *Chenciner*, *Flip+Neimark-Sacker*, *Fold+Neimark-Sacker*, *Resonance 1:1*, *Double Neimark-Sacker*, *Resonance 1:2*, *Resonance 1:3* and *Resonance 1:4*.

`h` and `f` were discussed in §2.5.1.

4.3.5 Adaptation

It is possible to adapt the problem while generating the Neimark-Sacker curve. This call updates the auxiliary variables used in the defining system of the computed branch. The bordering matrices V and W may require updating since they must at least be such that the matrix in (36) is nonsingular. Updating of V and $W$is done exactly as in the LimitPoint case. Updating of i_1, j_1, i_2, j_2 is done in such a way that the linearized system of (35) is as well - conditioned as possible.

5 Normal forms of codim-2 bifurcation points

Below we give normal forms to which the restriction of a generic map $g(x,\alpha) = f^{(K)}(x,\alpha)$ to the parameter-dependent center manifold can be transformed near the corresponding bifurcation by smooth invertible coordinate and parameter transformations. We only incorporate the parameter-dependent part if branch switching to local bifurcations is involved. The $\mathcal{O}$-symbol denotes higher order terms in phase-variables, the coefficients of which may also depend on parameters. But the qualitative picture is determined by the lowest order terms listed below. We refer to [33], Ch. 9, and [35, 36] for details, including explicit expressions for all critical normal form coefficients which are not repeated here. If a complex critical eigenvalue λ is involved, it is always assumed that $\lambda^\nu \neq 1$ for $\nu = 1, 2, 3, 4$.

5.1 CP (cusp)

The critical smooth normal form on the center manifold at a *cusp bifurcation* is

$$w \mapsto w + dw^3 + \mathcal{O}(|w|^4), \quad w \in \mathbb{R}, \tag{38}$$

where, generically, $d \neq 0$. Under this condition, a generic two-parameter unfolding of this singularity has two fold curves in the parameter plane which form a cuspidal wedge. For nearby parameter values, the map g has up to three fixed points that pairwise collide along the fold curves. In the direct product of the state and the parameter spaces, there is one smooth fold curve, so no branch switching is needed.
MatContM provides the normal form coefficient d.

5.2 GPD (generalized period doubling)

Near a *generalized flip* bifurcation the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$w \mapsto -(1+\beta_1)w + \beta_2 w^3 + c_2 w^5 + \mathcal{O}(|w|^6), \quad w \in \mathbb{R}, \tag{39}$$

where, generically, the coefficient $c_2 \neq 0$, while (β_1, β_2) are smooth functions of α which can serve as new unfolding parameters. The fixed point $w = 0$ of the map (39) exhibits a flip bifurcation for $\beta_1 = 0$. From the point $\beta = 0$, corresponding to the generalized flip bifurcation, a fold curve of double-period cycles emanates. The asymptotic expression for this curve in (39) is given by

$$(w, \beta_1, \beta_2) = (\varepsilon, -c_2\varepsilon^4 + \mathcal{O}(\varepsilon^5), -2c_2\varepsilon^2 + \mathcal{O}(\varepsilon^3)). \tag{40}$$

MatContM provides the normal form coefficient c_2. An example is given in §10.2, *Run 16.*

5.3 CH (Chenciner)

If $\mathrm{e}^{i\nu\theta_0} \neq 1$ for $\nu = 1, 2, \ldots, 6$, the critical smooth normal form on the center manifold at the *Chenciner bifurcation* can be written as

$$z \mapsto \mathrm{e}^{i\theta_0} z + c_1 z|z|^2 + c_2 z|z|^4 + \mathcal{O}(|z|^6), \quad z \in \mathbb{C}, \tag{41}$$

where $\Re(\mathrm{e}^{-i\theta_0} c_1) = 0$ but, generically, $\Re(\mathrm{e}^{-i\theta_0} c_2) + \frac{1}{2}\Im(\mathrm{e}^{-i\theta_0} c_1)^2 \neq 0$.
MatContM provides the coefficient $\Re(\mathrm{e}^{-i\theta_0} c_2) + \frac{1}{2}\Im(\mathrm{e}^{-i\theta_0} c_1)^2$ to check nondegeneracy.

5.4 R1 (resonance 1:1)

The restriction of the map at a 1:1 *resonance* to the corresponding center manifold can be written in the form

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \mapsto \begin{pmatrix} w_1 + w_2 \\ w_2 + a_1 w_1^2 + b_1 w_1 w_2 \end{pmatrix} + \mathcal{O}(\|w\|^3), \quad w \in \mathbb{R}^2. \tag{42}$$

Generically, a Neimark-Sacker bifurcation curve of the fixed point meets tangentially the fold bifurcation curve. The local branch switching problem is trivial here.
MatContM provides the sign of the coefficient $s = (b_1 - a_1)a_1$ which determines the stability of the bifurcating invariant curve.

5.5 R2 (resonance 1:2)

Near a 1:2 *resonance* the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$\begin{aligned} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \quad &\mapsto \quad \begin{pmatrix} -w_1 + w_2 \\ \beta_1 w_1 + (-1 + \beta_2) w_2 + C_1(\beta) w_1^3 + D_1(\beta) w_1^2 w_2 \end{pmatrix} \\ &+ \quad \mathcal{O}(\|w\|^4), \quad w \in \mathbb{R}^2. \end{aligned} \tag{43}$$

that depends on two control parameters (β_1, β_2). If $C_1 < 0$, then there is a Neimark-Sacker curve of fixed points of g with double period that emanates from the flip bifurcation curve $\beta_2 = 0$ of fixed points. It has the following asymptotic expression

$$(w_1^2, w_2, \beta_1, \beta_2) = \left(-\frac{1}{C_1}, 0, 1, \left(2 + \frac{D_1}{C_1} \right) \right) \varepsilon + \mathcal{O}(\varepsilon^2). \tag{44}$$

MatContM provides the normal form coefficients $[c, d] = [4C_1, -6C_1 - 2D_1]$, to check nondegeneracy conditions. Moreover the sign of the coefficient d determines the bifurcation scenario under generic pertubation, see [33], Ch 9. An example is given in §10.3, *Run 11*.

5.6 R3 (resonance 1:3)

At a 1:3 *resonance*, the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$z \mapsto (\mathrm{e}^{2i\pi/3} + \beta) z + B_1 \bar{z}^2 + C_1 z|z|^2 + \mathcal{O}(|z|^4), \quad z \in \mathbb{C}. \tag{45}$$

A generic unfolding of this singularity has a period-3 saddle cycle that does not bifurcate for nearby parameter values, although it merges with the primary fixed point as the parameters approach R3.
Note that the period-3 cycle becomes neutral near this bifurcation. Recall that a saddle cycle is called *neutral* if the corresponding fixed point has a pair of real multipliers with product 1. This singularity is important in analyzing global bifurcations of invariant manifolds of cycles. Moreover, the curve of neutral period-3 saddle cycles may turn into a true Neimark-Sacker bifurcation at R1 or R2. Therefore, we give here an asymptotic of this curve.

First we need a vector field for which the time-1 flow approximates the third iterate of the map, i.e.

$$\widetilde{g}(\eta,\widetilde{\beta}) = \widetilde{\beta}\eta + \bar{\eta}^2 + C_0\eta^2\bar{\eta} + \mathcal{O}(|\eta|^4), \tag{46}$$

where

$$\widetilde{\beta} = 3\mathrm{e}^{-2i\pi/3}\beta,\ z = \frac{1}{|B_1(\beta)|}\mathrm{e}^{i\arg(B_1(\beta))/3}\eta,\ C_0 = \frac{1}{3}\left(\mathrm{e}^{-2i\pi/3}C_1 - 1\right).$$

We write $C_0 = a + ib$ and for $\eta = \rho\mathrm{e}^{i\phi}$ the neutral saddle curve has the following asymptotic expression

$$(\rho,\phi,\beta_1,\beta_2) = \left(\varepsilon, s(\pi/6 - a\varepsilon/3), -2a\varepsilon^2, s\varepsilon - b\varepsilon^2\right) + \mathcal{O}(\varepsilon^3), \tag{47}$$

where $s = \pm 1$.
MatContM provides the coefficient $c_1 = e^{\frac{4\pi}{3}}C_1 - |B_1|^2$ which determines the bifurcation scenario. An example is given in §10.3, *Run 11*.

5.7 R4 (resonance 1:4)

Near a 1:4 *resonance* the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$z \mapsto (i+\beta)z + C_1(\beta)z^2\bar{z} + D_1(\beta)\bar{z}^3 + \mathcal{O}(|z|^4), \quad z \in \mathbb{C}. \tag{48}$$

For this bifurcation we do not only need this parameter-dependent normal form, but also an approximation of its 4th iterate by a unit-time shift along orbits of a vector field

$$\widetilde{g}(\eta,\widetilde{\beta}) = \widetilde{\beta}\eta + A_0(\beta)\eta^2\bar{\eta} + \bar{\eta}^3 + \mathcal{O}(|\eta|^4), \tag{49}$$

where $\eta \in \mathbb{C}$ and $\widetilde{\beta} = \tilde{\beta}_1 + i\tilde{\beta}_2, \tilde{\beta}_i \in \mathbb{R}$. Here the scaling

$$z = \frac{1}{\sqrt{|D_1(\beta)|}}\mathrm{e}^{i\arg(D_1(\beta))/4}\eta$$

is used and

$$A_0(\beta) = -i\frac{C_1(\beta)}{|D_1(\beta)|}.$$

Moreover, we have

$$\begin{pmatrix} \widetilde{\beta}_1 \\ \widetilde{\beta}_2 \end{pmatrix} = \begin{pmatrix} 0 & 4 \\ -4 & 0 \end{pmatrix}\begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}. \tag{50}$$

There are three possible branch switches for this bifurcation. Let $a = \Re(A_0(0))$ and $b = \Im(A_0(0))$. If $\Delta \equiv a^2 + b^2 - 1 > 0$, then there are two half-lines $l_{1,2}$ of a limit-point curve of cycles with four times the original period. If

$$|b| > \frac{(1+a^2)}{\sqrt{1-a^2}},$$

then there is a curve n_1 along which a cycle of four times the primary period exhibits a Neimark-Sacker bifurcation. Using $\eta = re^{i\phi}$ we have the following approximations

$$\begin{aligned} l_{1,2} : (r^2, \phi, \widetilde{\beta}_1, \widetilde{\beta}_2) =& \left(\varepsilon, \frac{1}{4}\arctan\left(\frac{ab \pm \sqrt{\Delta}}{b^2-1}\right) + \mathcal{O}(\varepsilon),\right. \\ & \left. \frac{-a\Delta \mp b\sqrt{\Delta}}{a^2+b^2}\varepsilon, \frac{-b\Delta \pm a\sqrt{\Delta}}{a^2+b^2}\varepsilon\right) + \mathcal{O}(\varepsilon^2) \\ n_1 : (r^2, \phi, \widetilde{\beta}_1, \widetilde{\beta}_2) =& \quad (\varepsilon + \mathcal{O}(\varepsilon^2), \operatorname{sign}(b)\arccos(a)/4 + \mathcal{O}(\varepsilon), \\ & -2a\varepsilon + \mathcal{O}(\varepsilon^2), -(b - \operatorname{sign}(b)\sqrt{1-a^2})\varepsilon + \mathcal{O}(\varepsilon^2)). \end{aligned} \tag{51}$$

Taking into account (50), we obtain expressions for β. If, in the formula for n_1, we replace $\operatorname{sign}(b)$ by $-\operatorname{sign}(b)$, then this gives the asymptotic for a neutral saddle singularity of the period-4 cycle.
MatContM provides the coefficient $A = -i\frac{C_1(0)}{|D_1(0)|}$ which determines the bifurcation scenario near the R_4 point. An example is given in §10.3, *Run 11*.

5.8 LPPD (fold –flip)

Near a *fold-flip* bifurcation, the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the normal form

$$\begin{aligned} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \quad \mapsto \quad & \begin{pmatrix} \beta_1 + (1+\beta_2)w_1 + a(\beta)w_1^2 + b(\beta)w_2^2 + c_1(\beta)w_1^3 + c_2(\beta)w_1w_2^2 \\ -w_2 + e(\beta)w_1w_2 + c_3(\beta)w_1^2w_2 + c_4(\beta)w_2^3 \end{pmatrix} \\ + \quad & \mathcal{O}(\|w\|^4), \quad w \in \mathbb{R}^2. \end{aligned} \tag{52}$$

A new branch predicted by (52) for a generic map g is a Neimark-Sacker of double period that exists if $be > 0$ and has the asymptotic expression

$$(x, y^2, \beta_1, \beta_2) = \left(-\frac{c_4}{e}, 1, -b, -\frac{2b + ec_2 - 2(a+e)c_4}{e}\right)\varepsilon + \mathcal{O}(\varepsilon^2). \tag{53}$$

MatContM provides the normal form coefficients $a(0)$ and $b(0)$. An example is given in §10.2, *Run 16*.

5.9 LPNS (fold – Neimark-Sacker)

For a *fold – Neimark-Sacker* bifurcation, the critical normal form on the center manifold is given by

$$\begin{pmatrix} w \\ z \end{pmatrix} \mapsto \begin{pmatrix} w + sz\bar{z} + w^2 + cx^3 \\ e^{i\theta_0}z + awz + bzw^2 \end{pmatrix} + \mathcal{O}(\|(w,z)\|^4), \quad (w,z) \in \mathbb{R} \times \mathbb{C}. \tag{54}$$

Depending on the coefficient values, several bifurcation scenarios are possible, which all involve only global phenomena.
MatContM provides the normal form coefficients s, a, b and c.

5.10 PDNS (flip – Neimark-Sacker)

Near a *flip – Neimark-Sacker* bifurcation, the restriction of the map g to the parameter-dependent center manifold is smoothly equivalent to the parameter-dependent normal form

$$\begin{pmatrix} w \\ z \end{pmatrix} \mapsto \begin{pmatrix} -(1+\beta_1)w + c_1(\beta)w^3 + c_2(\beta)w|z|^2 \\ e^{i\theta(\beta)}(1+\beta_2)z + c_3(\beta)w^2 z + c_4(\beta)z|z|^2 \end{pmatrix} + \mathcal{O}(\|(w,z)\|^4), \qquad (w,z) \in \mathbb{R}\times\mathbb{C}, \tag{55}$$

where $\theta(0) = \theta_0$. Besides global bifurcations, a Neimark-Sacker bifurcation curve of double period for g is rooted at $\beta = 0$; it is always present. The asymptotic expression of this curve is given by

$$(w^2, z, \beta_1, \beta_2) = \Big(1, 0, c_1, \mathrm{sign}(c_1)\Re(e^{-i\theta_0}c_3)\Big)\,\varepsilon + \mathcal{O}(\varepsilon^2). \tag{56}$$

MatContM provides the normal form coefficients $c_1(0), c_2(0), c_3(0)$ and $c_4(0)$.

5.11 NSNS (double Neimark-Sacker)

For a *double Neimark-Sacker bifurcation*, provided $l\theta_0 \neq j\theta_1$ for integer l and j with $l+j \leq 4$, the critical normal form on the center manifold is

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \mapsto \begin{pmatrix} e^{i\theta_0}z_1 + c_1 z_1|z_1|^2 + c_2 z_1|z_2|^2 \\ e^{i\theta_1}z_2 + c_3 z_2|z_1|^2 + c_4 z_2|z_2|^2 \end{pmatrix} + \mathcal{O}(\|z\|^4), \quad z \in \mathbb{C}^2. \tag{57}$$

Depending on the coefficient values, several bifurcation scenarios are possible in parameter-dependent unfoldings, which all involve only global phenomena. For some of them, one has to take into account fourth- and fifth-order terms.
MatContM provides the normal form coefficients $c_1(0), c_2(0), c_3(0)$ and $c_4(0)$.

6 Branch switching at codim-2 bifurcation points

The problem of branch switching is to specify one starting point near the curve from which the continuation code converges to a point on the curve.

Here we address the problem of branch switching at codim 2 bifurcation points of maps, when the emanating curve corresponds to a local bifurcation. These cases involve generalized flip, 1:2 resonance, 1:3 resonance, 1:4 resonance, fold-flip and flip-Neimark-Sacker bifurcations only. The asymptotic expressions for the new curves for the parameter-dependent normal form are given in Section 5. Combining this information with a parameter-dependent center-manifold reduction, leads to an initial prediction in state-parameter space.

6.1 Detection and switching graphs for codim-1 and codim-2 bifurcation points

In the detection graph Figure 4, we present all codim-1 and codim-2 bifurcation points that can be detected on curves of fixed points and codim-1 bifurcation curves. In the switching graphs Figure 5 and Figure 6, possible switchings at codim-1 and codim-2 bifurcation points are indicated graphically.

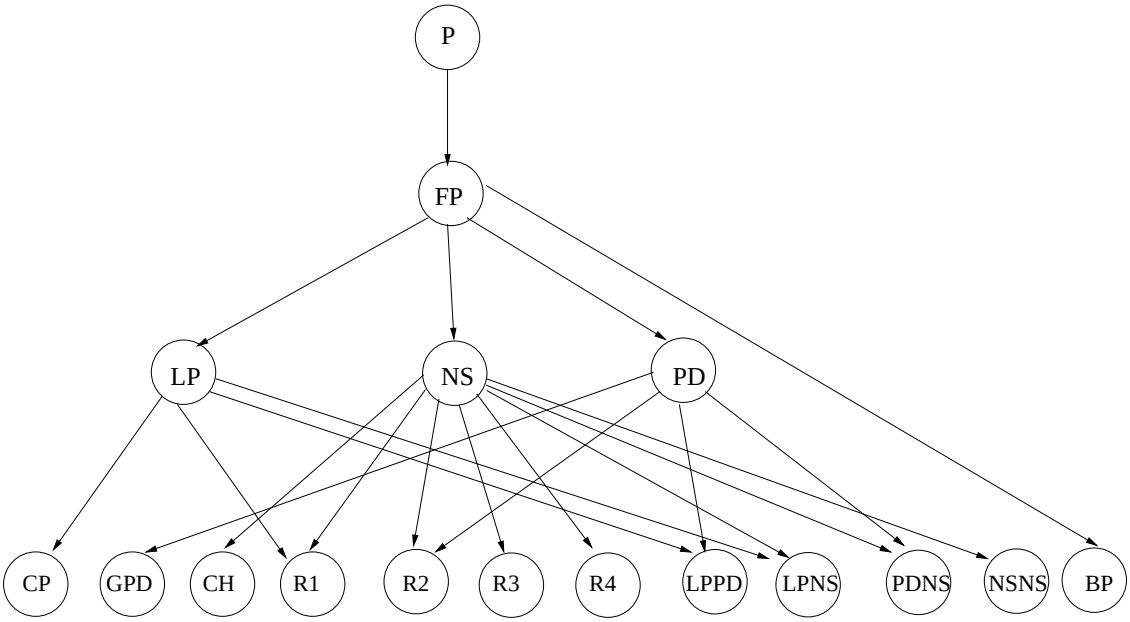


Figure 4: Detection graph.

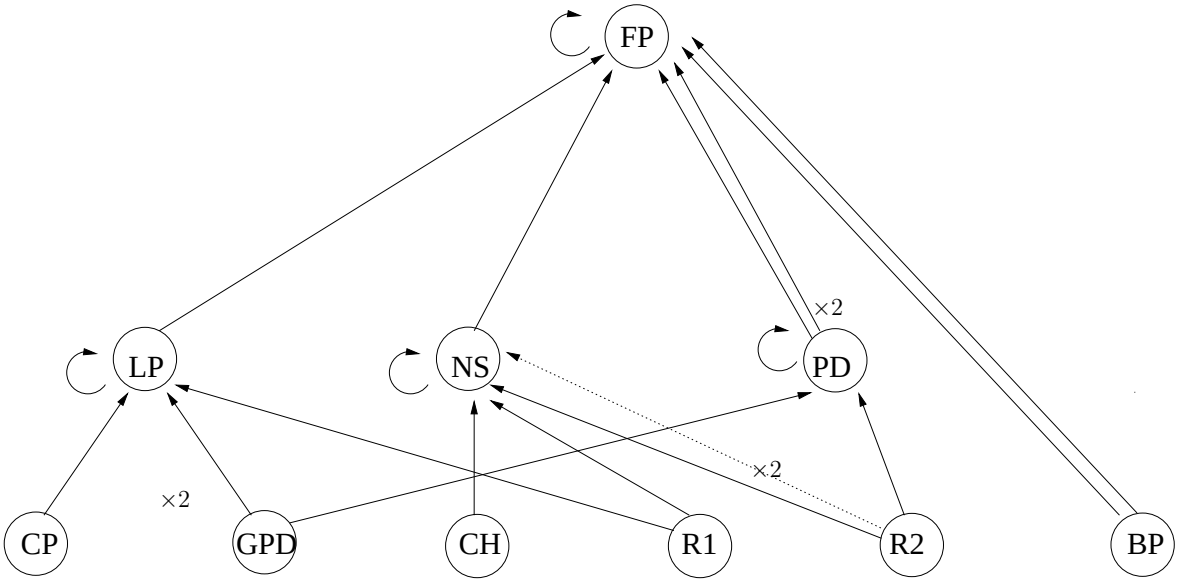


Figure 5: Switching graph 1: dashed lines indicate switching subject to constraints and ×2 indicates curve of double period.

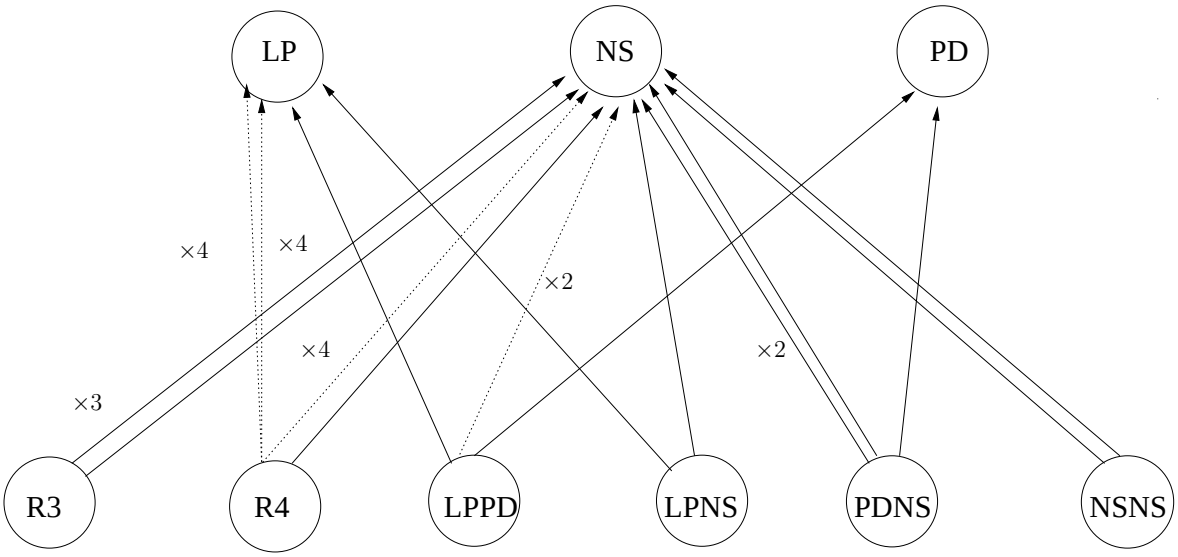


Figure 6: Switching graph 2: dashed lines indicate switching subject to constraints, ×2 and ×4 indicate curves of double and quadruple periods, respectively.

6.2 Initializations of branch switching

For each switch to a codim-1 curve an initializer m-function is constructed, the syntax is as follows :

```
[x0,v0] = init_GPD_LP2m(@mapfile, eps, x, p, ap, n)
[x0,v0] = init_R2_NS2m(@mapfile, eps, x, p, ap, n)
[x0,v0] = init_R3_NS3m(@mapfile, eps, x, p, ap, n)
[x0,v0] = init_R4_LP4m1(@mapfile, eps, x, p, ap, n)
[x0,v0] = init_R4_LP4m2(@mapfile, eps, x, p, ap, n)
[x0,v0] = init_R4_NSm4(@mapfile, eps, x, p, ap, n)
[x0,v0] = init_LPPD_NS2m(@mapfile, eps, x, p, ap, n)
[x0,v0] = init_PDNS_NS2m(@mapfile, eps, x, p, ap, n)
```

The arguments are

- mapfile An m-file containing the specifications of the map.

- eps The (positive) amplitude of the initial step.

- x The coordinates of the bifurcating fixed point.

- p The parameters at which the codim-2 bifurcation occurs.

- ap The active parameters which are used.

- n The number of iterates of the bifurcating fixed point.

In some cases it depends on the critical normal form coefficients whether branch switching is possible. If it is not, we stop. Otherwise, we specify a new coordinate $\tilde{x}$ and parameter $\tilde{p}$ and return $x0 = (\tilde{x}, \tilde{p})$. So, for example, for the generalized flip bifurcation $\tilde{x} = x + \epsilon q$

and $\tilde{p} = p - 2c_2v_{01}\epsilon^2 + (-c_2v_{10} + 2c_2^2v_{02})\epsilon^4$, where only the parameters specified by ap are updated. If the amplitude ϵ is small, it is enough to use only the leading terms of the asymptotic expressions.

7 Algorithmic and numerical details

In this section we consider the computation of the derivatives and tensor-vector products, which are not only needed for the continuation, but also for the computation of the critical normal form coefficients at codim 1 and 2 bifurcation points and for branch switching.

7.1 Recursive formulas for derivatives of iterates of maps

7.1.1 Derivatives with respect to phase variables

The iteration of (1) gives rise to a sequence of points

$$\{x_1, x_2, x_3, \ldots, x_{K+1}\},$$

where $x_{J+1} = f^{(J)}(x_1, \alpha)$ for $J = 1, 2, \ldots, K$. Suppose that symbolic derivatives of f up to order 5 can be computed at each point. We write

$$A(x_J)_{i,j} = \frac{\partial f_i}{\partial x_j}(x_J),\ B(x_J)_{i,j,k} = \frac{\partial^2 f_i}{\partial x_j \partial x_k}(x_J),\ C(x_J)_{i,j,k,l} = \frac{\partial^3 f_i}{\partial x_j \partial x_k \partial x_l}(x_J),$$

and similarly for $D(x_J)$ and $E(x_J)$.

We want to find recursive formulas for the derivatives of the composition (5), i.e. the coefficients of the multilinear functions in (13) that we now denote by $A^{(J)}, B^{(J)}$, and $C^{(J)}$ to indicate the iterate explicitly:

$$(A^{(J)})_{i,j} = \frac{\partial (f^{(J)}(x_1))_i}{\partial x_j},\ (B^{(J)})_{i,j,k} = \frac{\partial^2 (f^{(J)}(x_1))_i}{\partial x_j \partial x_k},\ (C^{(J)})_{i,j,k,l} = \frac{\partial^3 (f^{(J)}(x_1)_i}{\partial x_j \partial x_k \partial x_l},$$

and $D^{(J)}$ and $E^{(J)}$ are analogously defined. What follows is a straightforward application of the Chain Rule.

For $J = 1$ we have $A^{(1)} = A(x_1), B^{(1)} = B(x_1)$ and $C^{(1)} = C(x_1)$ and these are known. Now,

$$A^{(J)}_{i,j} = \sum_k \frac{\partial f_i}{\partial x_k}(f^{(J-1)}(x_1)) \frac{\partial (f^{(J-1)}(x_1))_k}{\partial x_j} = \sum_l A(x_J)_{i,k} A^{(J-1)}_{k,j} = A(x_J)A(x_{J-1})\ \ldots\ A(x_1). \tag{58}$$

we see that

$$(F(x,\alpha))_x = A(x^K)A(x^{K-1}) \cdots A(x^1) - I_n, \tag{59}$$

where $F(x, \alpha) = f^{(K)}(x, \alpha) - x$.
For the second order derivatives we first write $B^{(J)}$ once in coordinates

$$\begin{aligned} B^{(J)}_{i,j,k} &= \frac{\partial}{\partial x_j}\frac{\partial}{\partial x_k} f_i(f^{(J-1)}(x)) \\ &= \sum_{l,m} \frac{\partial^2 f_i}{\partial x_l \partial x_m}(x_J) \frac{\partial (f^{(J-1)})_m}{\partial x_j} \frac{\partial (f^{(J-1)})_l}{\partial x_k} + \sum_l \frac{\partial f_i}{\partial x_l}(x_J) \frac{\partial^2 (f^{(J-1)})_l}{\partial x_j \partial x_k}. \end{aligned}$$

For any two vectors q_1 and q_2, we can multiply the previous expression by $(q_1)_j(q_2)_k$ and sum over (k,l) to obtain

$$B^{(J)}(q_1,q_2) = B(x_J)(A^{(J-1)}q_1, A^{(J-1)}q_2) + A(x_J)B^{(J-1)}(q_1,q_2). \tag{60}$$

As $A(x_J)$ and $B(x_J)$ are known, (60) allows to compute the multilinear form $B^{(K)}(q_1,q_2)$ recursively.

Let $q_i, i = 1,2,3,4,5$, be given vectors. Multilinear forms with higher order derivatives can be computed with

$$\begin{array}{rl} C^{(J)}(q_1,q_2,q_3) = & C(x_J)(A^{(J-1)}q_1, A^{(J-1)}q_2, A^{(J-1)}q_3)+ \\ & B(x_J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3)^*+ \\ & A(x_J)(C^{(J-1)}(q_1,q_2,q_3)), \end{array} \tag{61}$$

where * means that all combinatorially different terms have to be included, i.e.,

$$\begin{array}{rcl} B(x_J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3)^* & = & B(x_J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3) + \\ & & B(x_J)(B^{(J-1)}(q_1,q_3), A^{(J-1)}q_2) + \\ & & B(x_J)(B^{(J-1)}(q_2,q_3), A^{(J-1)}q_1). \end{array}$$

For $D^{(J)}$ we get

$$\begin{array}{rl} D^{(J)}(q_1,q_2,q_3,q_4) = & D(x_J)(A^{(J-1)}q_1, A^{(J-1)}q_2, A^{(J-1)}q_3, A^{(J-1)}q_4)+ \\ & C(x_J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3, A^{(J-1)}q_4)^*+ \\ & B(x_J)(B^{(J-1)}(q_1,q_2), B^{(J-1)}(q_3,q_4))^*+ \\ & B(x_J)(C^{(J-1)}(q_1,q_2,q_3)), A^{(J-1)}q_4)^*+ \\ & A(x_J)D^{(J-1)}(q_1,q_2,q_3,q_4). \end{array} \tag{62}$$

Finally, for $E^{(J)}$ holds

$$\begin{array}{rl} E^{(J)}(q_1,q_2,q_3,q_4,q_5) = & E(x_J)(A^{(J-1)}q_1, A^{(J-1)}q_2, A^{(J-1)}q_3, A^{(J-1)}q_4, A^{(J-1)}q_5)+ \\ & D(x_J)(B^{(J-1)}(q_1,q_2), A^{(J-1)}q_3, A^{(J-1)}q_4, A^{(J-1)}q_5)^*+ \\ & C(x_J)(B^{(J-1)}(q_1,q_2), B^{(J-1)}(q_3,q_4), A^{(J-1)}q_5)^*+ \\ & C(x_J)(C^{(J-1)}(q_1,q_2,q_3), A^{(J-1)}q_4, A^{(J-1)}q_5)^*+ \\ & B(x_J)(C^{(J-1)}(q_1,q_2,q_3), B^{(J-1)}(q_4,q_5))^*+ \\ & B(x_J)(D^{(J-1)}(q_1,q_2,q_3,q_4))(A^{(J-1)}q_5)^*+ \\ & A(x_J)(E^{(J-1)}(q_1,q_2,q_3,q_4,q_5)). \end{array} \tag{63}$$

The multilinear forms $A^{(K)}(q_1)$, $B^{(K)}(q_1,q_2)$, $C^{(K)}(q_1,q_2,q_3)$, $D^{(K)}(q_1,q_2,q_3,q_4)$ and $E^{(K)}(q_1,q_2,q_3,q_4,q_5)$ are used in the computations of the normal form coefficients for codim 1 and codim 2 bifurcations of period-K cycles in §3.4 and §5, and also in the branch switching in §6.

7.1.2 Derivatives with respect to parameters

In the continuation of codimension 1 bifurcation curves (§4.1, §4.2, §4.3), we need derivatives of the form $\frac{\partial f^{(J)}}{\partial \alpha_k}$ and $\frac{\partial^2 f^{(J)}}{\partial \alpha_k \partial x}$ where α_k is a parameter. If enough symbolic derivatives of f are

available, then MATCONTM computes these expressions symbolically. The idea is as follows. Taking the derivative of (1) with respect to α_k, gives

$$\frac{\partial(f^{(J)}(x_1,\alpha))}{\partial\alpha_k} = \frac{\partial f}{\partial\alpha_k}(x_J,\alpha) + \frac{\partial f}{\partial x}(x_J,\alpha)\frac{\partial(f^{(J-1)}(x_1,\alpha))}{\partial\alpha_k}, \tag{64}$$

which is recursively computable. Also mixed derivatives, which are necessary for continuation and branch switching, can be found recursively:

$$\frac{\partial^2(f^{(J)}(x_1,\alpha))}{\partial\alpha_k\partial x} = \frac{\partial^2 f}{\partial\alpha_k\partial x}(x_J,\alpha) + \frac{\partial^2 f}{\partial x^2}(x_J,\alpha)\frac{\partial(f^{(J-1)}(x_1,\alpha))}{\partial\alpha_k}. \tag{65}$$

In fact, the recursion is not applied to (65) itself, but to its product with a fixed vector.

This is sufficient for all continuations of fixed points and their codimension 1 bifurcations. It is also sufficient for all cases of branch switching from codimension 2 points, except for the case of generalized flip. For this case, we fall back to a finite difference approximation. Since it is only used in the prediction step for which high accuracy is not needed, this seems acceptable.

7.2 Recursive formulas for derivatives of the defining systems for continuation

For the continuation of fixed points and cycles we need the derivatives of (1) which can be computed from (59) and (64). Now, we consider the derivatives of g (as defined in (23)) with respect to z, a state variable or parameter. The flip and Neimark-Sacker cases can be handled in a similar way. Let M be the matrix in (24). By taking derivatives of (24) with respect to z we obtain

$$M\begin{bmatrix} v_z \\ g_z \end{bmatrix} + \begin{bmatrix} A_z^{(K)} & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} v \\ g \end{bmatrix} = 0. \tag{66}$$

Using (26) we obtain

$$g_z = -w^T(A^{(K)})_z v. \tag{67}$$

If z represents one of the state variables, then

$$g_{x_i} = -\langle w, B^{(K)}(e_i, v)\rangle, \tag{68}$$

as computed in section 7.1. When z is a parameter α_k we can write

$$g_{\alpha_k} = \sum_{J=1}^{K} C_J, \tag{69}$$

where

$$C_J = -w^T f_x(x_K)\ \cdots\ (f_x(x_J))_{\alpha_k} f_x(x_{J-1})\ \cdots\ f_x(x_1)v \tag{70}$$

where $J = 1,\ldots,K$. In this expression

$$(f_x(x_J))_{\alpha_k} = [f_x(f^{(J)}(x^1,\alpha))]_{\alpha_k} = f_{x\alpha}(x_J,\alpha) + B(x_J)T_J \tag{71}$$

where T_J is a vector, that can be recursively defined by

$$T_J = f_{\alpha_k}(x_{J-1},\alpha) + A(x_{J-1})T_{J-1}, \quad T_1 = 0. \tag{72}$$

Summarizing, for the computation of g_α we need to compute $f_x, f_{\alpha_k}, f_{xx}, f_{x\alpha_k}$ in all iteration points $x_1, \ldots, x_K$, and given these compute T_J for $J = 1, \ldots K$. Then

$$C_J = -w^T A(x_K) \ \cdots \ (f_{x\alpha_k}(x_J) + B(x_J)T_J)A(x_{J-1}) \ \cdots \ A(x_1)v \tag{73}$$

and g_{α_k} is computed via (69).

7.3 Computing the vector-Hessian-vector and Hessian-vector products

To define the Jacobian system of fold, flip and Neimark-Sacker continuations we need to compute terms of the form vector-Hessian-vector and Hessian-vector where Hessian is a $n \times n \times n$ tensor and vector is a n-vector. These can be computed symbolically by using the recursive formulas derived in Section 7.2. Computation of vector-Hessian-vector by using (68) and (69) are implemented in the files `lpvecthessvect` and `lpvecthesspvect`. These files are used in fold continuation and contained in the folder `LimitPointMap`. The same computations are implemented in `pdvecthessvect` and `pdvecthesspvect` for flip continuation and `nsvecthessvect` and `nsvecthesspvect` for Neimark-Sacker continuation. Furthermore, in switching of some codim-2 bifurcation points, we need to compute expressions of the form Hessian-vector symbolically, where Hessian is a $n \times n \times n$ tensor w.r.t state variables and parameters. This computation can be done by using the recursive formula (69) and implemented in `lphesspvect`, `pdhesspvect` and `nshesspvect` and are contained in the folders `LimitPointMap`, `PeriodDoublingMap`, `NeimarkSackerMap`, respectively.

7.4 Finite difference approximation of directional derivatives

For a general discussion of directional derivatives we refer to [33], §10.3.4, where it is shown that all computations can be reduced to computing expressions of the form

$$f_x q, \quad f_{xx} qq, \quad f_{xxx} qqq, \quad f_{xxxx} qqqq \quad , \quad f_{xxxxx} qqqqq$$

For the first order derivative we approximate:

$$f_x q \cong \tfrac{f(x+hq)-f(x-hq)}{2h} \tag{74}$$

For the second order derivative, we have

$$f_{xx} qq \cong \tfrac{f(x+hq)-2f(x)+f(x-hq)}{h^2} \tag{75}$$

Similarly for the third order derivative, we have

$$f_{xxx} qqq \cong \tfrac{f(x+3hq)-3f(x+hq)+3f(x-hq)-f(x-hq)}{8h^3} \tag{76}$$

For the fourth order derivative we have:

$$f_{xxxx} qqqq \cong \tfrac{f(x+4hq)-4f(x+2hq)+6f(x)-4f(x-2hq)+f(x-4hq)}{16h^4} \tag{77}$$

Finally, for the fifth order derivative

$$f_{xxxxx} qqqqq \cong \tfrac{f(x+5hq)-5f(x+3hq)+10f(x+hq)-10f(x-hq)+5f(x-3hq)-f(x-5hq)}{32h^5} \tag{78}$$

In the software the default values of the increment h for the first, the second, the third, the fourth and the fifth order derivatives are defined as follows:

- $Increment = (\epsilon_m)^{\frac{1}{3}}$
- $hessIncrement = (\epsilon_m)^{\frac{1}{4}}$
- $tens3Increment = (\epsilon_m)^{\frac{1}{5}}$
- $tens4Increment = (\epsilon_m)^{\frac{1}{6}}$
- $tens5Increment = (\epsilon_m)^{\frac{1}{7}}$

where ϵ_m is the machine precision; we use $\epsilon_m = 10^{-15}$. However, the *Increment* can be adjusted by the user by setting *cds.options.Increment*. The increments of the higher-order derivatives are then adapted accordingly.

7.5 Using automatic differentiation

As an alternative to symbolic derivatives (SD) and finite differences (FD) for computing normal form coefficients, we have implemented automatic differentiation (AD) techniques, see e.g. [26], into MATCONTM, to compute derivatives of an iterated map, w.r.t state variables. More precisely, in the computation of the multilinear functions (13), a drawback of using the recursive expressions (58), (60), (61), (62) and (63) is the nonlinear growth of elapsed time with increasing iteration number. To remedy this drawback, we use AD for high iteration numbers of the map. If symbolic derivatives are not available, then we always use AD in the computation of normal form coefficients.
The user has the possibility to switch off the use of AD or to link its use to the iteration number of the map. The syntax for this is as follows:

```
options=contset(options,'AutDerivative',i);
```

where i is 1 or 0, decides whether AD is to be used or not, respectively. The default value for i is 1.
The command:

```
options=contset(options,'AutDerivateIte',i);
```

where i is an integer number, instructs the software to use AD in the computation of normal form coefficients if the iteration number of the map equals or exceeds i (default $i = 24$). As a rule, it can be advised to lower i for increasing phase space dimension.

7.6 Multilinear forms

The files which are specific to the directional multilinear forms are stored in the directory `MultilinearForms`. These files are used for computation of the multilinear functions A, B, C, D and E in (13). We use these multilinear forms to compute the normal form coefficients of the codimension-1 and codimension-2 bifurcations as well as for defining some test functions for the codimension-2 bifurcation points and branch switching. The computations are done numerically with finite directional differences or using symbolic derivatives of the original map depending on the *mapfile* or AD. The directory `MultilinearForms` contains the following *m*-files:

- `multilinear1.m` computes Aq where q is a vector. In the case of symbolic derivatives the computations are done by calling the file `multilinear1sym.m`, which uses the recursive formula (58).
- `multilinear2.m` computes $B(q_1 q_2)$ where q_1 and q_2 are vectors. In the case of symbolic derivatives the computations are done by calling the file `multilinear2sym.m`, which uses the recursive formula (60).
- `multilinear3.m` computes $C(q_1, q_2, q_3)$ where q_1, q_2 and q_3 are vectors. In the case of symbolic derivatives the computations are done by calling the file `multilinear3sym.m`, which uses the recursive formula (61).
- `multilinear4.m` computes $D(q_1, q_2, q_3, q_4)$ where q_1, q_2, q_3 and q_4 are vectors. In the case of symbolic derivatives the computations are done by calling the file `multilinear4sym.m`, which uses the recursive formula (62).
- `multilinear5.m` computes $E(q_1, q_2, q_3, q_4, q_5)$ where q_1, q_2, q_3,q_4 and q_5 are vectors. In the case of symbolic derivatives the computations are done by calling the file `multilinear5sym.m`, which uses the recursive formula (63).
- The files `nf_LPm.m`, `nf_PDm.m`, `nf_NSm.m`, `nf_CPm.m`, `nf_DPDm.m`, `nf_CHm.m`, `nf_LPPDm.m`, `nf_LPNSm.m`, `nf_PDNSm.m`, `nf_NSNSm.m`, `nf_R1m.m`, `nf_R2m.m`, `nf_R3m.m` and `nf_R4m.m` compute the critical normal form coefficients of codim-1 and codim-2 bifurcation points `LP` (fold), `PD` (flip), `NS` (Neimark-Sacker), `CP` (Cusp), `GPD` (generalized flip), `CH` (Chenciner), `LPPD` (fold+flip), `LPNS` (fold+Neimark-Sacker), `PDNS` (flip+Neimark-Sacker), `NSNS` (double Neimark-Sacker), `R1` (Resonance1:1), `R2` (Resonance1:2), `R3` (Resonance1:3), and `R4` (Resonance1:4), respectively.

8 Numerical continuation of connecting orbits of maps

The accurate computation of orbits connecting fixed points of an iterated map, and the study of associated topological properties have long been recognized as an important problem both in the theory of nonlinear dynamical systems and in a variety of applied problems, e.g. in models for economical, biological, and physical phenomena. Indeed, as discovered by Poincaré and Birkhoff, such orbits may generate rich dynamics. For example, an orbit that connects a hyperbolic fixed point to itself (a homoclinic orbit) generically implies the existence of an infinite number of periodic orbits nearby, see [57, 44, 56] and tutorial presentations in [28, 61, 50]. As discovered in [20, 21, 27], the appearance of a pair of such homoclinic orbits is accompanied by an infinite sequence of fold and period-doubling bifurcations of periodic orbits, for more details see [48, 51], as well as [33]. Moreover, since a homoclinic orbit of a planar map belongs to the intersection of the stable and the unstable invariant curves of a saddle fixed point, such orbits can be involved in the destruction of a closed invariant curve which is born, for example, at a Neimark-Sacker bifurcation [47, 54, 55]. This destruction mechanism has been studied in [2, 7].

8.1 Continuation of heteroclinic connections

We consider the J-th iterate of a map at some parameter value α as follows:

$$x \mapsto f^J(x, \alpha), \quad f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n, \tag{79}$$

where

$$f^J(x,\alpha) = \underbrace{f(f(f(\cdots f}_{J \text{ times}}(x,\alpha),\alpha),\alpha),\alpha).$$

A sequence $(x_k)_{k\in\mathbb{Z}}$ is called a *connecting orbit* of the map $f^J(\cdot,\alpha)$ at $\alpha = \bar{\alpha}$ if

$$\begin{aligned} &\lim_{k\to-\infty} x_k = x_{-\infty} \\ &f^J(x_k,\bar{\alpha}) = x_{k+1}, \quad \text{for all} \quad k \in \mathbb{Z} \\ &\lim_{k\to+\infty} x_k = x_{+\infty} \end{aligned} \tag{80}$$

It is called *homoclinic* if $x_{-\infty} = x_{+\infty}$ and *heteroclinic* otherwise. From a geometrical point of view, the connecting orbit lies in the intersection of the unstable manifold $W^u_{-\infty}$ of $x_{-\infty}$ and the stable manifold $W^s_{+\infty}$ of $x_{+\infty}$. A connecting orbit is *regular* if $x_{-\infty}$ and $x_{+\infty}$ are hyperbolic and the stable manifold $W^u_{+\infty}$ and the unstable manifold $W^s_{-\infty}$ have transversal intersections at x_k for all $k \in \mathbb{Z}$.

Degenerate cases occur when either the orbit loses transversality or one of its fixed points becomes nonhyperbolic. In the latter case the unstable and center-stable manifolds have a transversal intersection, which produces a connecting orbit with a singular endpoint. In the simplest case there is precisely one multiplier 1 or -1, or one conjugate pair of multipliers of $f^J(x,\alpha)$ on the unit circle. This gives us the *saddle-fold, saddle-flip* or *saddle-Neimark-Sacker* connecting orbits, respectively, see e.g. [29, 4]. We will not deal with these cases but concentrate on nontransversality.

The numerical problem, for a regular heteroclinic connection between hyperbolic fixed points x_1 and x_N of (79), is that of finding a solution $(x_k)_{k=1,2,\ldots,N}$ of the following system [5]:

$$\begin{aligned} &x_1 = f^J(x_1,\alpha), \\ &x_{k+1} = f^J(x_k,\alpha), \quad k = 2,\ldots,N-2 \\ &x_N = f^J(x_N,\alpha) \end{aligned} \tag{81}$$

such that $(x_k)_{k=2..,N-1}$ leave x_1 along its unstable manifold and enter x_N along its stable manifold. These requirements are then substituted by *projection boundary conditions* which place x_2 and x_{N-1} into the corresponding tangent spaces [5].

We use an improved algorithm for locating and continuing connecting orbits, which includes an algorithm for the continuation of invariant subspaces (CIS) as described in [10, 14]. Assume the eigenvalues of $(f^J(x_1,\alpha))_x$ and $(f^J(x_N,\alpha))_x$ are ordered, respectively, as follows:

$$|\lambda^U_n| \le \ldots \le |\lambda^U_{n_U+1}| < 1 < |\lambda^U_1| \le \ldots \le |\lambda^U_{n_U}|$$

$$|\lambda^S_1| \le \ldots \le |\lambda^S_{n_S}| < 1 < |\lambda^S_{n_S+1}| \le \ldots \le |\lambda^S_n|$$

The algorithm requires the evaluation of various projections associated with the eigenspaces of $(f^J(x_1,\alpha))_x$ and $(f^J(x_N,\alpha))_x$. These projections are constructed using the real Schur factorizations.

$$(f^J(x_1,\alpha))_x = Q_1 R_1 Q_1^T, \quad (f^J(x_N,\alpha))_x = Q_2 R_2 Q_2^T.$$

where Q_1, R_1, Q_2 and R_2 are $n \times n$-matrices.

The first factorization is chosen so that the first n_U columns $q^U_1,\ldots,q^U_{n_U}$ of Q_1 form an orthonormal basis of the right invariant subspace S_1 of $(f^J(x_1,\alpha))_x$, corresponding to $\lambda^U_1,\ldots,\lambda^U_{n_U}$ and the remaining $n - n_U$ columns $q^U_{n_U+1},\ldots,q^U_n$ of Q_1 form an orthonormal

basis of the orthogonal complement $S_1^{\perp}$. Similarly, the first n_S columns $q_1^S, \ldots, q_{n_S}^S$ of Q_2 form an orthonormal basis of the right invariant subspace S_N of $(f^J(x_N, \alpha))_x$, corresponding to $\lambda_1^S, \ldots, \lambda_{n_S}^S$ and the remaining $n - n_S$ columns $q_{n_S+1}^S, \ldots, q_n^S$ of Q_2 form an orthonormal basis of the orthogonal complement $S_N^{\perp}$.

The problem of heteroclinic connections is to find a connection $\{x_n\}$ with:

- Stationary state conditions for the initial fixed point:

$$f^J(x_1, \alpha) - x_1 = 0, \tag{82}$$

- The iteration conditions

$$f^J(x_k, \alpha) - x_{k+1} = 0, \quad k = 2, 3, \ldots, N-2, \tag{83}$$

- Stationary state conditions for the final fixed point:

$$f^J(x_N, \alpha) - x_N = 0, \tag{84}$$

- The left boundary conditions

$$(x_2 - x_1)^T . q_{n_{U+i}}^U = 0, \quad i = 1, \ldots, n - n_U, \tag{85}$$

- The right boundary conditions

$$(x_{N-1} - x_N)^T . q_{n_{S+i}}^S = 0, \quad i = 1, \ldots, n - n_S \tag{86}$$

A regular zero of a system of equations (82), (83), (84), (85) and (86) corresponds to a transversal heteroclinic orbit with hyperbolic fixed points and with $n_U + n_S = n$. Thus, a zero for this system can be continued in one parameter.

In the computational process the conditions in (85) and (86) imply that at each step of the continuation we need to access the unstable and stable eigenspaces of the map (79) at the fixed points x_1 and x_N, respectively. It is not efficient to recompute these spaces from scratch in each continuation step. We use an algorithm for continuing the invariant subspaces S_1 and S_2, based on the Riccati equations, see [30]. Contrary to [10, 14], our algorithm is purely based on linear algebra arguments.

8.1.1 Heteroclinic initialization

To initialize the continuation of a heteroclinic connection to the saddle fixed points x_1 and x_N, one can start from an initial set of points as transversal intersections of invariant manifolds (to be discussed in §9) at x_0 and x_1. The initializer is called as follows:

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_Het\_Het}(@\texttt{mapfile}, \texttt{C}, \texttt{p}, \texttt{ap}, \texttt{J})$$

The input argument `mapfile`, is the map to be used and `C` is a matrix whose N columns contain coordinates of points in the intersection of invariant manifolds at x_1 and x_N, the first column corresponding to x_1 and the last to x_N. Next, `p` is the vector containing the starting values of parameters, `ap` is the index of the active parameter and `J` is the iteration number of the map. The output of `init_Het_Het` contains a vector `x0` with the continuation variables and an empty vector `v0`. Part of the output is also written in the global structures `cds` and `hetds`.

8.1.2 Output of the continuation of a heteroclinic connection

The continuation of a heteroclinic connection is performed by the command

$$[\texttt{x}, \texttt{v}, \texttt{s}, \texttt{h}, \texttt{f}] = \texttt{cont}(@\texttt{heteroclinic}, \texttt{x0}, [], \texttt{opt})$$

This call returns :
`x` and `v`: points and their tangent vectors along the heteroclinic connection, respectively.
The array `s` contains information about the computed singular points with the following fields:

s.index index of the singularity point in x.
s.label label of the singularity, may be `BP`, `LP`.
s.data values of the active test functions and user functions, in a `LP` point also the entries of a uni

`h` was discussed in §2.5.1. `f` contains for each computed point the number of orbit points in phase space (so it is constant vector).

8.1.3 Adaptation

At each continuation point a basis for the unstable eigenspace of x_1 and for its orthogonal complement are computed. However, these bases are not orthogonal. To restore orthogonality we adapt the basis for the unstable eigenspace from time to time using the *singular value decomposition* (SVD). By using a similar procedure we adapt the basis for the stable eigenspace of x_N and for its orthogonal complement.

8.2 Continuation of homoclinic connections

Assume that the eigenvalues of $(f^J(x_1, \alpha)_x$ are ordered as follows:

$$|\lambda_1| \leq \ldots \leq \ |\lambda_k| < 1 < |\lambda_{k+1}| \leq \ldots \leq |\lambda_n|$$

The procedure to continue a homoclinic connection to x_1 is similar to the procedure used in §8.1. The algorithm now requires the evaluation of two projections associated with the eigenspaces of $(f^J(x_1, \alpha)_x$. These projections are constructed using the real Schur factorizations.

$$(f^J(x_1, \alpha))_x = Q_1 R_1 Q_1^T, \quad (f^J(x_1, \alpha))_x = Q_2 R_2 Q_2^T$$

where Q_1, Q_2, R_1 and R_2 are $n \times n$-matrices.

The first factorization has been chosen so that the first k columns $q_1^S, \ldots, q_k^S$ of Q_1 form an orthonormal basis of the right invariant subspace S_1 of $(f^J(x_1, \alpha)_x$, corresponding to $\lambda_1, \ldots, \lambda_k$ and the remaining $n-k$ columns $q_{k+1}^U, \ldots, q_n^U$ of Q_1 form an orthonormal basis of the orthogonal complement $S_1^\perp$. Similary the first $l = n-k$ columns $q_1^U, \ldots, q_l^U$ of Q_2 form an orthonormal basis of the right invariant subspace U_1 of $(f^J(x_1, \alpha))_x$, corresponding to $\lambda_{k+1}, \ldots, \lambda_n$ and the remaining $n - l = k$ columns $q_{l+1}^U, \ldots, q_n^U$ of Q_2 form an orthonormal basis of the orthogonal complement $U_1^\perp$.

The problem of a *homoclinic* connection is to find a connection $\{x_m\}_{m=1,\ldots,N}$ with

- Stationary state condition
$$f^J(x_1, \alpha) - x_1 = 0, \tag{87}$$
- The iteration conditions
$$f^J(x_m, \alpha) - x_{m+1} = 0, \quad m = 2, 3, \ldots, N-2, \tag{88}$$

- The left boundary conditions

$$(x_2 - x_1).q^U_{k+i}(\alpha) = 0, \quad i = 1, \ldots, n-k \tag{89}$$

- The right boundary conditions

$$(x_{N-1} - x_1).q^S_{l+i}(\alpha) = 0, \quad i = 1, \ldots, n-l \tag{90}$$

A regular zero of a system of equations (87), (88), (89) and (90) corresponds to a transversal homoclinic orbit. Thus, a zero for this system can be continued in one parameter.

In the continuation process the conditions in (89) and (90) imply that we need to access the stable and unstable eigenspaces of the map (79) at the fixed points x_1 at each step of the continuation.

8.2.1 Homoclinic initialization

To initialize the continuation of a homoclinic connection to the saddle fixed point x_1, one should start from an initial set of points as transversal intersections of invariant manifolds at x_1. The initializer is called as follows:

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_Hom\_Hom}(@\texttt{mapfile}, \texttt{C}, \texttt{p}, \texttt{ap}, \texttt{J})$$

The input argument `mapfile` is the map to be used and `C` is a matrix with N columns where the first column contains the coordinates of the saddle point x_1 an the other columns contain the coordinates of points in the intersection of invariant manifolds at x_1. Next, `p` is the vector containing the starting values of the parameters, `ap` is the index of the active parameter and `J` is the iteration number of the map. The output of `init_Hom_Hom` contains a vector `x0` with the continuation variables and an empty vector `v0`. Part of the output of `init_Hom_Hom` is stored in the global structures `cds` and `homds`.

8.2.2 Output of continuation of a homoclinic connection

The continuation of a homoclinic connection is performed by the command

$$[\texttt{x}, \texttt{v}, \texttt{s}, \texttt{h}, \texttt{f}] = \texttt{cont}(@\texttt{homoclinic}, \texttt{x0}, [], \texttt{opt})$$

This call returns :
`x` and `v`: points and their tangent vectors along the homoclinic connection, respectively.
The array `s` contains information about the computed singular points with the following fields:

s.index	index of the singularity point in x.
s.label	label of the singularity, may be `BP` and `LP`.
s.data	values of the active test functions and user functions,

`h` was discussed in §2.5.1. `f` contains for each point the number of orbit points in phase space (so it is a constant vector).

8.2.3 Adaptation

At each continuation point a basis for the unstable eigenspace of x_1 and for its orthogonal complement are computed. However, these bases are not orthogonal. To restore orthogonality we adapt the basis for the unstable eigenspace from time to time using the *singular value decomposition* (SVD). By using a similar procedure we adapt the basis for the unstable eigenspace of x_1 and for its orthogonal complement.

8.3 Continuation of heteroclinic and homoclinic tangencies

Let $F(X,\alpha)=0$ be the defining system of the heteroclininc connection, then a heteroclinic tangency satisfies the following conditions:

$$\left\{\begin{array}{rcl} F(X,\alpha) & = & 0, \\ det(F_X(X,\alpha)) & = & 0, \end{array}\right. \tag{91}$$

which is a system of $K_1 = n(N-1)+n_U(n-n_U)+n_S(n-n_S)+2n-n_U-n_S+1$ equations in a $K_2 = nN+n_U(n-n_U)+n_S(n-n_S)+\#ap$-dimensional space with coordinates (X,α). Here $X=(x_1,\ldots,x_N,Y_U,Y_S,ap)$, Y_U is an auxiliary $(n-n_U)\times n_U$ matrix which is used in the adaptation of the bases for the unstable manifold at x_1 and its orthogonal complement (see the Riccati equations [30]), and Y_S is an auxiliary $(n-n_S)\times n_S$ matrix which is used in the adaptation of the bases for the stable manifold at x_N and its orthogonal complement (again, see the Riccati equations [30]).
If $n_U+n_S=n$ and $\#ap=2$, then (91) defines a continuation problem. This system is natural from of a theoretical perspective but may lead to numerical scaling problems. If the Jacobian has eigenvalues of large magnitude, then these eigenvalues contribute to the determinant (which is the product of all eigenvalues) and may make it difficult to satisfy the defining equations to a desired tolerance. The larger the system, the worse this problem becomes. Thus we seek alternate defining equations that avoid calculation of the determinant. Bordered matrices allow us to find a substitute function of the determinant.

We define a curve of heteroclinic tangencies by the following system

$$\left\{\begin{array}{rcl} F(X,\alpha) & = & 0, \\ g(X,\alpha) & = & 0, \end{array}\right. \tag{92}$$

where $g(X,\alpha)$ is computed as the last component of the solution vector in the K_1-dimensional bordered system:

$$\left(\begin{array}{cc} F_X(X,\alpha) & b \\ c^T & 0 \end{array}\right)\left(\begin{array}{c} v \\ g \end{array}\right)=\left(\begin{array}{c} 0_{(K_1-1)} \\ 1 \end{array}\right), \tag{93}$$

for suitable vectors $b,c\in\mathbb{R}^{K_1-1}$.

If c is close to the nullvector of $F_X(X,\alpha)$ and b is close to the nullvector of $F_X^T(X,\alpha)$, then the matrix

$$M=\left(\begin{array}{cc} F_X(X,\alpha) & b \\ c^T & 0 \end{array}\right) \tag{94}$$

is nonsingular at (X,α) and (93) has a unique solution. In practical computations, c and b are approximations of the null vectors of $F_X(X,\alpha)$ and $F_X^T(X,\alpha)$, respectively.

In the continuation of heteroclinic tangencies b and c are computed in the curve initializer *init_HetT_HetT* and stored in the fields *hetTds.b* and *hetTds.c* of the global variable *hetTds*.

The vectors b and c must be adapted during the continuation of hetereoclinic tangencies to keep the matrix M nonsingular.

8.3.1 Initialization of heteroclinic and homoclinic tangencies

To initialize the continuation of a heteroclinic or homoclinic tangency, one should start from a limit point found on a heteroclinic or homoclinic connection, respectively. The initializers for heteroclinic and homoclinic tangencies are called as follows, respectively:

$$[\texttt{x0},\texttt{v0}] = \texttt{init\_HetT\_HetT}(@\texttt{mapfile},\texttt{X},\texttt{nphase},\texttt{nu},\texttt{ns},\texttt{p},\texttt{ap},\texttt{J})$$

and

$$[\texttt{x0}, \texttt{v0}] = \texttt{init\_HomT\_HomT}(@\texttt{mapfile}, \texttt{X}, \texttt{nphase}, \texttt{nu}, \texttt{ns}, \texttt{p}, \texttt{ap}, \texttt{J})$$

The input arguments:

- `mapfile`, is the map to be used.
- `X` is a vector that contains the coordinates of the orbit points at the detected tangency.
- `nphase` is the dimension of the state space.
- `nu` and `ns` are the dimensions of the unstable and stable eigenspaces of the start point and end point, respectively.
- `p` and `ap` contain the current values of the parameters and the indices of the 2 active parameters, respectively.
- J is the iteration number of the map.

8.3.2 Output of a continuation of heteroclinic or homoclinic tangencies

The continuations of heteroclinic and homoclinic tangencies are performed by the commands

$$[\texttt{x}, \texttt{v}, \texttt{s}, \texttt{h}, \texttt{f}] = \texttt{cont}(@\texttt{heteroclinicT}, \texttt{x0}, [], \texttt{opt})$$

and

$$[\texttt{x}, \texttt{v}, \texttt{s}, \texttt{h}, \texttt{f}] = \texttt{cont}(@\texttt{homoclinicT}, \texttt{x0}, [], \texttt{opt})$$

respectively. This call returns :
`x` and `v`: points and their tangent vectors along the heteroclinic and homoclinic connections, respectively.

The array `s` contains information about the computed singular points with the following fields:

s.index	index of the singularity point in x.
s.label	label of the singularity.
s.data	values of the active test functions and user functions,

`h` was discussed in §2.5.1. `f` contains for each point the number of orbit points in phase space (so it is a constant vector).

9 Invariant manifolds

Invariant manifolds give information about the global structure of phase space. For example, a codimension 1 manifold in 2D space may separate several basins of attraction. Invariant manifolds are also used to simplify dynamical systems. The phase portrait near the manifold may be trivial, so restricting the dynamical system to the manifold effectively reduces the dimension of the system.

Our main motivation for computing stable and unstable manifolds of a saddle point is the role that they play in the computation of connecting orbits. Intersections of stable and unstable manifolds may form homoclinic or heteroclinic tangles. Stable and unstable manifolds are global objects that cannot normally be found analytically and, hence, must be computed

numerically. These manifolds must be grown from local knowledge, for example from linear information near a fixed point. We concentrate here on the simplest case that these manifolds are one-dimensional.

We recall some definitions, mostly to fix the notation. We consider (79) when $n = 2$ and assume that f has a fixed point $x_0 = f^J(x_0)$ and that f is differentiable in a neighborhood of x_0, but may not have a unique inverse. The fixed point x_0 of f is a saddle if the Jacobian matrix $D(f^J)(x_0)$ has at least one stable eigenvalue and one unstable eigenvalue, and no eigenvalue with modulus 1. The stable manifold theorem [50] guarantees that there exist local stable and unstable manifolds $W^s_{loc}(x_0)$ and $W^u_{loc}(x_0)$ tangent at x_0 to the stable and unstable eigenspaces $E^s(x_0)$ and $E^u(x_0)$, respectively. The stable manifold $W^s(x_0)$ of x_0 is defined as the set of points that converge to x_0 under forward iteration of f,

$$W^s(x_0) = \left\{ x \in R^2 : f^J(x) \to x_0 \ \text{ as } \ J \to \infty \right\}. \tag{95}$$

Similarly, the unstable manifold $W^u(x_0)$ of x_0 consists of points that converge to x_0 under backward iteration of the map f. In terms of forward iterates, this is defined as

$$W^u(x_0) = \left\{ x \in R^2 : \exists \{q_k\}, q_0 = x \ \text{ and } \ f^J(x_{q+1}) = q_k, \ \text{ and } \ \lim_{k\to\infty} q_k = x_0 \right\}. \tag{96}$$

The global stable manifold $W^s(x_0)$ can also be defined as the union of the successive pre-images of $W^s_{loc}(x_0)$. However, if multiple inverses exist, then all pre-images, even if disjoint from the main branch, are part of the stable manifold. Hence the stable manifold may or may not be simply connected in phase space.

9.1 Growing stable and unstable manifolds

In MatContM only one-dimensional stable and unstable manifolds are computed. We use (a slightly improved version of) the algorithm for computing the global one-dimensional unstable manifold of a saddle point of a map as described in [31]. The computation of one-dimensional stable manifolds of a planar map at a saddle point is described in [18]. An extension of this method to higher dimensions was proposed by C. Bruschi. We use a slightly improved variant of this extension. See also [46].

In general, the accuracy of the computation of a manifold is controlled locally by two numbers, namely the distance δ between two consecutive points and the angle α between two consecutive chord vectors. Depending on their sizes a newly found point is accepted or rejected and the input initial value of δ is halved, doubled or left unchanged in the next computational step.

9.1.1 Directory structure

The directory `InvManifolds` contains (at least) 7 files. The computation of a manifold starts with a call to `init_FPm_1Dman.m` in which the global structure *opt_man* and the curve description file `man_ds` are created. Next, a call to `contman.m` controls the actual computation of the manifold. Since the details of the computation differ in the stable and unstable cases, `contman.m` further calls `Smanifold.m` or `Umanifold.m`, depending on the case.
In the case of 2D maps with a saddle fixed point it is further possible to compute the intersection points between a computed stable and a computed unstable manifold, and hence to find homoclinic or heteroclinic connections. This is achieved by a call to either `findintersections.m`

or `Projectie2.m`. The output of this routine is a cell array in which each cell is an array whose columns constitute an orbit of the map. An application is given in §10.5.2.
The directory `InvManifolds` contains two further files `Smanifoldfile.m` and `Umanifoldfile.m` which at present are obsolete and will not be used. They were written to swap files in the case where the computed manifolds are too big to be kept in the internal Matlab memory. Advances in the Matlab software have made this procedure redundant but it is still possible to use `Smanifoldfile.m` and `Umanifoldfile.m` by a switch in the code in `contman`.

9.1.2 Options

When growing a one-dimensional manifold one has to start from the *options* structure which may already exist or can be created with `contset`.
`options = contset`
initializes *options* with empty field values, cf. Section 2.5.3. A call to `Init_FPm_1Dman.m` creates a new structure *opt_man* with some pre-assigned fields taken from *options* or given default values. These fields can be overwritten by the user; other fields can be added and set by the user. During the actual computation of the 1D manifolds by a call to `contman.m` fields can be changed and new fields can be added. The following fields will eventually appear:

deltaMin the minimal distance between two consecutive points on the computed manifold (MinStepsize in *options*, default: $1e-10$)

deltaMax the maximal distance between two consecutive points on the computed manifold (MaxStepsize in *options*, default: $0.1\sqrt{2}$. It is implicitly assumed that the minimal distance is not larger than the maximal distance but this is not checked by the code.

deltak the initially proposed distance between two consecutive points on the computed manifold (InitStepsize in *options*, default: $1e-4$). It is implicitly assumed that it is not smaller than the minimal distance and not larger than the maximal distance but this is not checked by the code.

nmax maximum number of computed points along the manifold (MaxNumPoints in *options*, default: 5000)

eps a threshold value used in the control of Newton-Raphson iterations in the computation of stable manifolds; not used in the computation of unstable manifolds (FunTolerance in *options*, default: $1e-6$). Decreasing the value of ϵ increases the accuracy of the computations but makes them slower and can lead to failure.

NwtMax maximum number of Newton iterations to locate a zero of a function (MaxNewtonIters in *options*, not used in the case of unstable manifolds, default: 10)

Niterations the iteration number of the map. It is mandatory that this field be set by the user. Otherwise an error will be declared. To preserve the direction, the iteration number is doubled if the leading eigenvalue is negative.

function can be either 'UManifold' or 'SManifold'. This field can be set by the user in the case of 2D maps. Default in this case is 'UManifold'. If the dimension is higher than 2, the code decides which type of one-dimensional manifold can exist and sets the field accordingly.

direction a unit right eigenvector corresponding to the leading (stable or unstable) eigenvalue. This field is set by the code. See also the field **distanceInit**.

Arc maximal arclength of the computed manifold. (Default: Inf).

alphaMax maximum allowed value for the angle α (default: pi).

alphaLowMax a stricter maximum value for α (default: pi/4).

deltaAlphaMax maximum value allowed for the product $\alpha\delta$.(default: 1e-3)

deltaAlphaLowMax a stricter maximum value for the product $\alpha\delta$ (default: 1e-4).

distanceInit distance from the saddle fixed point to the first point on the manifold, in the **direction** chosen by the code. A negative value forces the code to start in the opposite direction. (default: 1e-4).

searchListLength this is an obsolete field provided for the use of `Smanifoldfile.m` and `Umanifoldfile.m` (default: 20).

file this is an obsolete field, in which 1 indicates that `Smanifoldfile.m` or `Umanifoldfile.m` is used (default: 0).

The rules for accepting found points and for doubling or halving δ are as follows.

- A found point is accepted if and only if $\delta \leq$ deltaMax and $\alpha \leq$ alphaMax and $\delta\alpha \leq$ deltaAlphaMax.
- If a found point is accepted and in addition $\alpha \leq$ alphaLowMax and $\alpha\delta \leq$ deltaAlphaLowMax and $2\delta \leq$ deltaMax, then δ is doubled as initial guess for the next try.
- If a found point is not accepted and $\frac{\delta}{2} \geq$ deltaMin then δ is halved as initial guess for the next try.
- If a found point is not accepted and $\frac{\delta}{2} <$ deltaMin then the computation of the manifold is halted.

9.1.3 Output of the growing of an invariant manifold

An invariant manifold is grown by calling:

$$[\mathtt{a}, \mathtt{l}] = \mathtt{growman}(\mathtt{optM}); \tag{97}$$

The formal input is the structure `optM` which is used as the `opt_man` structure described in §9.1.2. We note that the structure `man_ds` is global in `growman.m` and must be set beforehand by calling `Init_FPm_1DMan`. This call returns :
`a`: an array in which the entries of each column are the coordinates of a point on the manifold.
`l`: the arclength of the manifold.

10 Examples and applications

10.1 A truncated normal form map

10.1.1 The map and some analytical normal form coefficients

In this example we consider the two - dimensional map, introduced in [33], §9.9, (unfolding of an *R2* point to which it reduces for $\beta_1 = \beta_2 = 0$)

$$M_{TN} : \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} \longmapsto \begin{pmatrix} -1 & 1 \\ \beta_1 & -1+\beta_2 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} + \begin{pmatrix} 0 \\ C\xi_1^3 + D\xi_1^2\xi_2 \end{pmatrix} \tag{98}$$

For all parameter values, this map has a trivial fixed point $(0,0)^T$. If (ξ_1, ξ_2) is a nontrivial fixed point then we have:

$$\xi_2 = 2\xi_1, \quad \xi_2 = \beta_1\xi_1 + (-1+\beta_1)\xi_2 + C\xi_1^3 + D\xi_1^2\xi_2 \tag{99}$$

It is easy to see that if

$$\frac{4-(\beta_1+2\beta_2)}{C+2D} > 0,$$

then nontrivial real fixed points $(\xi_1, 2\xi_1)$ exist and are given by

$$\xi_1 = \pm\sqrt{\frac{4-(\beta_1+2\beta_2)}{C+2D}}, \xi_2 = 2\xi_1. \tag{100}$$

If

$$\frac{4-(\beta_1+2\beta_2)}{C+2D} = 0,$$

then these points collide with a trivial fixed point. If this happens with β_1 or β_2 as a free parameter in a continuation of trivial fixed points, then clearly we have a pitchfork bifurcation of fixed points.
The characteristic equation of the Jacobian in the trivial fixed point is:

$$\lambda^2 + (2-\beta_2)\lambda + 1 - \beta_1 - \beta_2 = 0. \tag{101}$$

We first note that the product of the two multipliers is 1 if and only if $\beta_1+\beta_2 = 0$. In particular, `NS` points can only be found if $\beta_1 + \beta_2 = 0$. In this case, $\Delta = (2-\beta_2)^2 - 4 = \beta_2(\beta_2 - 4)$. So we have true `NS` points if $\beta_2 \in]0,4[, \beta_1 = -\beta_2$; we have neutral saddles if $\beta_2 \notin [0,4], \beta_1 = -\beta_2$.

In particular we consider the following three special cases of the `NS` bifurcation:

$$\begin{array}{ll} (i) & \beta_1 = -1, \beta_2 = 1, (\theta = \frac{2\pi}{3}) \\ (ii) & \beta_1 = -2, \beta_2 = 2, (\theta = \frac{\pi}{2}) \\ (iii) & \beta_1 = -3, \beta_2 = 3, (\theta = \frac{\pi}{3}) \end{array} \tag{102}$$

We note that cases (i) and (ii) are cases with a strong resonance.

Also, it is easy to see that (101) has a root -1 if and only if $\beta_1 = 0$. The other root then is $-1+\beta_2$. We will also consider the case :

$$(iv)\beta_1 = 0, \beta_2 = 1. \tag{103}$$

One can obtain analytically the normal form coefficients. The results are as follows:

- in the case of (i), i.e. $\theta = \frac{2\pi}{3}$: $c = -\frac{1}{8}(6C + 4D)$
- in the case of (ii), i.e. $\theta = \frac{\pi}{4}$: $c = -\frac{1}{12}(6C + 6D)$
- in the case of (iii), i.e. $\theta = \frac{\pi}{3}$: $c = -\frac{1}{16}(6C + 8D)$
- in the case of (iv), i.e. $\theta = \pi$: $b = -C$

10.1.2 Numerical continuation of trivial and nontrivial fixed points

Theoretically computed values of the normal form coefficients can now be checked numerically when continuing the fixed point curve. In the *mapfile* (cf. §2.6) the order of state variables and parameters is (ξ_1, ξ_2) and (β_1, β_2, C, D), respectively. For illustration purposes we defined two user functions, namely $\beta_2 - 2$ with label $'B2\ \ '$ and $\beta_2 - 0.5$ with label $'B3\ \ '$.
First we continue the fixed point curve numerically to detect the `NS` point in case (i) in *Run 1*, where we use the mapfile that uses symbolic derivatives (cf. §2.6).

```
>> init
>> global opt cds fpmds
>> ap=2; p=[-1;0;1;1];
>> opt = contset;
>> n=1;
>> [x0,v0]=init_FPm_FPm(@Tnfmap,[0;0], p, ap, n);
>> opt=contset(opt,'MaxNumPoints',50);
>> opt=contset(opt,'Singularities',1);
>> opt = contset(opt,'Multipliers',1);
>> opt=contset(opt,'Userfunctions',1);
>> UserInfo(1).label='B2 ';
>> UserInfo(1).name='Beta2';
>> UserInfo(1).state=1;
>> UserInfo(2).label='B3 ';
>> UserInfo(2).name='Beta3';
>> UserInfo(2).state=1;
>> opt=contset(opt,'UserfunctionsInfo',UserInfo);
>> [x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = B3  , x = ( 0.000000 0.000000 0.500000 )
label = NS  , x = ( 0.000000 0.000000 1.000000 )
normal form coefficient of NS = -1.250000e+000
label = B2  , x = ( 0.000000 0.000000 2.000000 )
label = BP  , x = ( 0.000000 0.000000 2.500000 )
elapsed time  = 0.7 secs
npoints curve = 50
```

This test is run by typing *Tnfmap1* in the command window.
Our theoretical outcome is confirmed by the numerical value for d obtained in *Run 1*. Indeed, in this case the theoretically obtained value of the normal form coefficient c is

$$c = -\frac{1}{8}(6C + 4D) = -1.25$$

since $C = D = 1$.

By (100) the nontrivial fixed points collide to a trivial fixed point when

$$4 - (\beta_1 + 2\beta_2) = 0 \tag{104}$$

The fixed parameter in *Run 1* is $\beta_1 = -1$, this implies that in a BP $\beta_2 = 2.5$ in (104). This confirms the result in *Run 1* concerning the BP point.

The Jacobian is given by:

$$[(M_{TN})_x - I|(M_{TN})_{\beta_2}] = \begin{pmatrix} -2 & 1 & 0 \\ \beta_1 & -2+\beta_2 & 0 \end{pmatrix} \tag{105}$$

If $\beta_1 = -1$ and $\beta_2 = 2.5$, then this reduces to:

$$[(M_{TN})_x - I|(M_{TN})_{\beta_2}] = \begin{pmatrix} -2 & 1 & 0 \\ -1 & 0.5 & 0 \end{pmatrix} \tag{106}$$

Clearly $[(M_{TN})_x - I|(M_{TN})_{\beta_2}]$ is rank deficient as expected.

Now we compute the new branch in the BP point of *Run 1*; we refer to this *Run 2*:

```
>> global x1 v1 s1  opt cds  fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
 >>>>> Branch switching at BP >>>>>>>
>> xx2=x1(1:2,s1(5).index);p1=p;p1(fpmds.ActiveParams)=x1(3,s1(5).index);
>> opt=contset(opt,'Backward',0);
>>opt=contset(opt,'MaxNumPoints',50);
>>[x2,v2]=init_BPm_FPm(@Tnfmap,xx2,p1,s1(5),0.01,1);
>>[x21,v21,s21,h21,f21]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = PD, x = ( 0.377964 0.755929 2.285714 )
normal form coefficient of PD = 4.392157e+000
elapsed time  = 0.6 secs
npoints curve = 50
>> cpl(x21,v21,s21,[3 1]);
>>opt=contset(opt,'Backward',1);;
>> [x22,v22,s22,h22,f22]=cont(@fixedpointmap,x2,[],opt);
first point found
tangent vector to first point found
label = BP, x = ( -0.000000 -0.000000 2.500000 )
label = PD, x = ( -0.377964 -0.755929 2.285714 )
normal form coefficient of PD = 4.392157e+000
elapsed time  = 0.9 secs
npoints curve = 50
>> cpl(x22,v22,s22,[3 1])
```

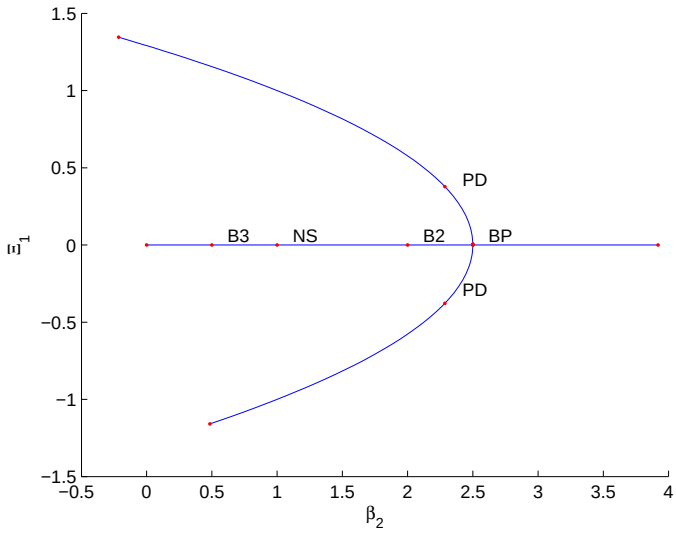


Figure 7: Continuation of trivial and nontrivial fixed points of M_{TN} in (β_2, ξ_1) space .

This test is run by typing *Tnfmap2* in the command window.
The branch in *Run 2* is a nontrivial one and we remark that for the singular points $\xi_2 = 2\xi_1$ holds. In fact the curve of nontrivial fixed points in (100) in (β_2, ξ_1) space is a parabola. A picture of the continued trivial fixed point of *Run 1* and nontrivial fixed points computed in *Run 2* is given in Figure 7.

In *Run 3* we continue a fixed point curve to detect the `NS` point in case (ii) :

```
>> global opt cds fpmds
>> ap=2; p=[-2;0;1;1];
>> opt = contset;
>> [x0,v0]=init_FPm_FPm(@Tnfmap,[0;0], p, ap,1);
>> opt=contset(opt,'MaxNumPoints',300);
>> opt=contset(opt,'Singularities',1);
>> opt=contset(opt,'Backward',0);
>> opt = contset(opt,'Multipliers',1);
>>[x3,v3,s3,h3,f3]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = NS   , x = ( 0.000000 0.000000 2.000000 )
normal form coefficient of NS = -1.000000e+000
label = BP   , x = ( 0.000000 0.000000 3.000000 )
elapsed time  = 1.1 secs
npoints curve = 300
>> cpl(x3,v3,s3,[3 1])
```

This test is run by typing *Tnfmap3* in the command window.
Again the numerically obtained value for the normal form coefficient d in *Run 3* confirms the theoretical result. Indeed, the theoretical value is

$$d = -\frac{1}{12}(6C + 6D) = -1$$

since $C = D = 1$.
By (100) we have a BP point if $4 - (\beta_1 + 2\beta_2) = 0$. By subsituting $\beta_1 = -2$, we get $\beta_2 = 3$. This confirms the result in *Run 3* concerning the BP point.

Now we perform branch switching in this point in *Run 4*:

```
>> global x3 v3 s3 opt cds  fpmds
>> opt = contset;
>> xx2=x3(1:2,s3(3).index);p1=p;p1(fpmds.ActiveParams)=x3(3,s3(3).index);
>> opt=contset(opt,'Backward',0);
>> opt=contset(opt,'MaxNumPoints',300);
>> opt=contset(opt,'Singularities',1);
>> opt=contset(opt,'Multipliers',1);
>> [x2,v2]=init_BPm_FPm(@Tnfmap,xx2,p1,s3(3),0.001,1);
>> [x41,v41,s41,h41,f41]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = PD, x = ( 0.534523 1.069045 2.571429 )
normal form coefficient of PD = 3.733333e+000
elapsed time  = 1.4 secs
npoints curve = 300
>> cpl(x41,v41,s41,[3 1 2]);
>>opt=contset(opt,'Backward',1);;
>> [x42,v42,s42,h42,f42]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = BP, x = ( -0.000000 -0.000000 3.000000 )
label = PD, x = ( -0.534523 -1.069045 2.571429 )
normal form coefficient of PD = 3.733333e+000
elapsed time  = 1.5 secs
npoints curve = 300
>> cpl(x42,v42,s422,[3 1 2])
```

This test is run by typing *Tnfmap4* in the command window.
The BP point is the same as in *Run 3*, and the PD point on the new branch satisfies $\xi_2 = 2\xi_1$.
A picture of the new branch computed in *Run 4* is given in Figure 8.

In *Run 5* we continue a fixed point curve to detect the NS point in case (*iii*) :

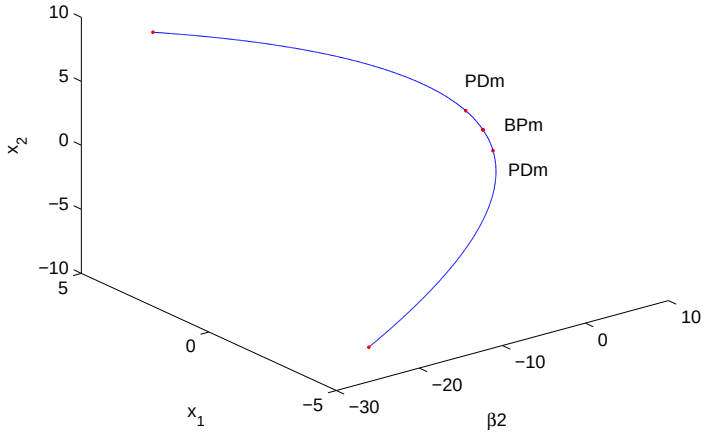


Figure 8: The fixed points curve of the second iterate in (β_2, x_1, x_2) space.

```
>> global opt cds fpmds
>> ap=2; p=[-3;0;1;1];
>> opt = contset;
>> [x0,v0]=init_FPm_FPm(@Tnfmap,[0;0], p, ap,1);
>> opt=contset(opt,'MaxNumPoints',300);
>> opt=contset(opt,'Singularities',1);
>> opt = contset(opt,'Multipliers',1);
>>[x5,v5,s5,h5,f5]=cont(@fixedpointmap,x0,[],opt);
first point found
tangent vector to first point found
label = NS, x = ( 0.000000 0.000000 3.000000 )
normal form coefficient of NS = -8.750000e-001
elapsed time  = 1.4 secs
npoints curve = 300
>> cpl(x5,v5,s5,[3 1])
```

This test is run by typing *Tnfmap5* in the command window.
Here also the numerically obtained value confirms the theoretical result
$c = -\frac{1}{16}(6C + 8D) = -0.875$ where $C = D = 1$.
Since the normal form coefficient in *Run 5* is negative, the invariant curves nearby the `NS` point must be stable. In fact the characteristic polynomial (101) when $\beta_1 = -3$, is

$$\lambda^2 + (2 - \beta_2)\lambda + 4 - \beta_2 = 0 \tag{107}$$

The multipliers for β_2 nearby 3 are

$$\lambda_{1,2} = -\frac{\beta_2 - 2}{2} \pm i\sqrt{3 - \frac{\beta_2^2}{4}}$$

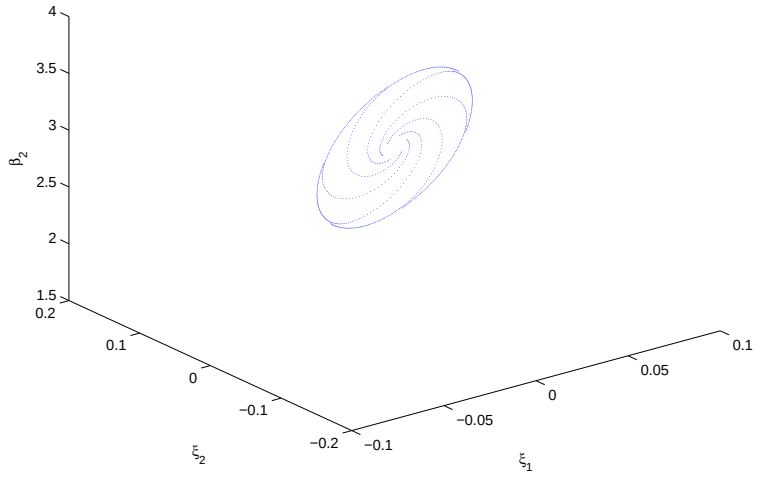


Figure 9: Stable invariant curve of M_{TN} started from $\xi_1 = \xi_2 = 0.01$ for $\beta_2 = 2.99$.

Also

$$|\lambda_{1,2}^2| = 4 - \beta_2$$

So the fixed point of M_{TN} is stable for $\beta_2 > 3$ and unstable for $\beta_2 < 3$, i.e. the invariant curve is stable when $\beta_2 < 3$ and unstable when $\beta_2 > 3$. A picture of the stable invariant curve nearby the `NS` point is given in Figure 9. It was created by simulation of M_{TN} for the parameter values indicated in Figure 9.

The next *Run 6* will detect the `PD` point in case (iv):

```
>> global  opt cds fpmds
>> ap=1; p=[-1;1;1;1];
>> opt = contset;
>> [x0,v0]=init_FPm_FPm(@Tnfmap,[0;0], p, ap,1);
>> opt=contset(opt,'MaxNumPoints',300);
>> opt=contset(opt,'Singularities',1);
>> opt = contset(opt,'Multipliers',1);
>>[x6,v6,s6,h6,f6]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = NS, x = ( 0.000000 0.000000 -1.000000 )
normal form coefficient of NS = -1.250000e+000
label = PD, x = ( 0.000000 0.000000 0.000000 )
normal form coefficient of PD = -1
label = BP, x = ( 0.000000 0.000000 2.000000 )
```

```
elapsed time  = 1.3 secs
npoints curve = 300
```

This test is run by typing *Tnfmap6* in the command window.
Clearly the result of the continuation in *Run 6* is consistent with the theoretical statement for case (iv), that is $b = -2C = -2$ since $C = 1$.
By (100) we have a BP point when $4 - (\beta_1 + 2\beta_2) = 0$. Since $\beta_2 = 1$ in *Run 6*, the BP point must be found for $\beta_1 = 2$. This confirms the result in *Run 6* concerning the BP point.

Now we compute the curve of fixed points of the second iterate in the PD point of *Run 6*. We call this *Run 7*:

```
>> global x6 v6 s6  opt cds  fpmds
>> opt = contset;
 >>>>>  switching at PD >>>>>>>
>> xx2=x6(1:2,s6(3).index);p1=p;
>> p1(fpmds.ActiveParams)=x6(3,s6(3).index);
>> opt=contset(opt,'Backward',1);
>> opt=contset(opt,'MaxNumPoints',300);
>> opt=contset(opt,'Singularities',1);
>> [x2,v2]=init_PDm_FP2m(@Tnfmap,xx2,p1,s6(3),0.01,1);
>> opt = contset(opt,'Multipliers',1);
>> [x7,v7,s7,h7,f7]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = BP, x = ( -0.000000 0.000000 -0.000000 )
label = NS, x = ( -0.577350 0.000000 -0.333333 )
Neutral saddle
label = BP, x = ( -0.707107 -0.000000 -0.500000 )
elapsed time  = 2.0 secs
npoints curve = 300
```

This test is run by typing *Tnfmap7* in the command window.
The first BP point in *Run 7* is the PD point in *Run 6*. The second BP point corresponds with $\beta_1 = -0.5$ and is clearly not a fixed point of M_{TN}. For the parameters values in *Run 7*, we have:

$$M_{TN} : \begin{pmatrix} -\frac{\sqrt{2}}{2} \\ 0 \end{pmatrix} \longmapsto \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} -\frac{\sqrt{2}}{2} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ 0 \end{pmatrix} \tag{108}$$

And

$$M_{TN} : \begin{pmatrix} \frac{\sqrt{2}}{2} \\ 0 \end{pmatrix} \longmapsto \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\sqrt{2}}{2} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -\frac{\sqrt{2}}{2} \\ 0 \end{pmatrix} \tag{109}$$

So indeed M_{TN}^2 maps the point $(-\frac{\sqrt{2}}{2}, 0)^T$ to itself.

We further remark that in *Run 7* the trivial fixed point is stable for negative values of β_1 close to 0 and unstable for positive values of β_1 close to zero. Also, the normal coefficient of the PD point is negative.

Further computations show that the multipliers of M_{TN} and M_{TN}^2 in the PD point $x = (0, 0, 0)$ are $(-1, 0)^T$ and $(1, 0)^T$ respectively.

A nearby point on the curve of fixed points of the second iterate is $x = (-0.0129710, 0.0000000, -0.0001683)$. For the same parameter value the fixed point of the map is stable, as could be expected from the sign of the normal form coefficient in the PD point in *Run 6*. The multipliers of M_{TN} and M_{TN}^2 in the same point are $(-1.0003363, 0.0005046)^T$ and $(1.0006728, 0.0000003)^T$ respectively. So the fixed points of the second iterate are unstable, as could also be expected from the sign of the normal form coefficient in the PD point in *Run 6*.

10.2 A Leslie-Gower competition model

10.2.1 The model and its fixed points

The roots of the present model can be found in [37, 38, 17]. Roughly speaking, it was found in biological experiments that two species of flour beetles can coexist under strong competition for the same food. This was rather unexpected at the time and several models were built to explain this phenomenon. One of the ideas in [17] and [58] is to use an age-structured competition model. For general background we refer to [6]; the model that we use is a four - dimensional map M_{LG} (110) with 14 parameters described in [58]. It is a Leslie/Gower competition model for the interaction between the juveniles (j) and adults (a) of one species of the flour beetle Tribolium and the juveniles (y) and adults (z) of another species for the same food.

$$M_{LG} : \begin{pmatrix} j \\ a \\ y \\ z \end{pmatrix} \longmapsto \begin{pmatrix} j_+ \\ a_+ \\ y_+ \\ z_+ \end{pmatrix} = \begin{pmatrix} \dfrac{b_1 a}{(1 + c_{jj} j + c_{ja} a + c_{jy} y + c_{jz} z)} \\ (1 - \mu_j) j + (1 - \mu_a) a \\ \dfrac{b_2 z}{(1 + c_{yj} j + c_{ya} a + c_{yy} y + c_{yz} z)} \\ (1 - \mu_y) y + (1 - \mu_z) z \end{pmatrix} \tag{110}$$

Each species has its own juvenile recruitment rate $b_1 > 0$, $b_2 > 0$, juvenile death rate μ_j and μ_y, and adult death rate μ_a and μ_z. For biological reasons we have

$$0 < \mu_j, \mu_a, \mu_y, \mu_z < 1. \tag{111}$$

The other coefficients $c_{jj}, c_{ja}, c_{jy}, c_{jz}$ and $c_{yj}, c_{ya}, c_{yy}, c_{yz}$ describe the competition. They are all strictly positive. By assumption, competition does not affect the adults of either species. In the present study, as in [58], we will study the influence of the coefficients c_{yj} and c_{jy} on the behavior of M_{LG} in a case where all other parameters are fixed. In other words, we study the influence of the competition between juveniles if all other parameters are fixed.
The fixed points (j^*, a^*, y^*, z^*) of the map (110) satisfy:

$$\begin{aligned} a^* &= \tfrac{1-\mu_j}{\mu_a} j^* \\ z^* &= \tfrac{1-\mu_y}{\mu_z} y^*. \end{aligned} \tag{112}$$

As a consequence j^* and a^* are either both zero or both nonzero. Similarly, y^*, z^* are both zero or both nonzero. The trivial vector $(0, 0, 0, 0)^T$ is a fixed point of (110). It can be checked

easily by analytical means that it is always unstable.
For each set of parameters there is a unique 'horizontal' fixed point, i.e. a fixed point of the form $(j^*, a^*, 0, 0)$ which is given by:

$$j^* = \frac{b_1(1-\mu_j) - \mu_a}{\mu_a c_{jj} + c_{ja}(1-\mu_j)} \tag{113}$$

$$a^* = \frac{1-\mu_j}{\mu_a} j^* = \frac{b_1(1-\mu_j)^2 - \mu_a(1-\mu_j)}{\mu_a(\mu_a c_{jj} + c_{ja}(1-\mu_j))} \tag{114}$$

with $(j^*, a^*) > 0$ (i.e. j^*, a^* are biologically meaningful) iff

$$b_1 \frac{1-\mu_j}{\mu_a} > 1 \tag{115}$$

Similarly, there is a unique 'vertical' fixed point of the form $(0, 0, y^*, z^*)$ given by

$$y^* = \frac{b_2(1-\mu_y) - \mu_z}{\mu_z c_{yy} + c_{yz}(1-\mu_y)} \tag{116}$$

$$z^* = \frac{1-\mu_y}{\mu_z} y^* = \frac{b_2(1-\mu_y)^2 - \mu_z(1-\mu_y)}{\mu_z(\mu_z c_{yy} + c_{yz}(1-\mu_y))} \tag{117}$$

This point is biologically meaningful if $(y^*, z^*) > 0$, i.e. iff

$$b_2 \frac{1-\mu_y}{\mu_z} > 1 \tag{118}$$

The general form of the Jacobian matrix of (110) is:

$$\begin{pmatrix} \frac{-b_1 c_{jj} a}{\beta_1^2} & \frac{b_1\beta_1 - b_1 a c_{ja}}{\beta_1^2} & \frac{-b_1 c_{jy} a}{\beta_1^2} & \frac{-b_1 c_{jz} a}{\beta_1^2} \\ 1-\mu_j & 1-\mu_a & 0 & 0 \\ \frac{-b_1 c_{yj} z}{\beta_2^2} & \frac{-b_2 c_{ya} z}{\beta_2^2} & \frac{-b_1 c_{yy} z}{\beta_2^2} & \frac{b_2\beta_2 - b_2 z c_{yz}}{\beta_2^2} \\ 0 & 0 & 1-\mu_y & 1-\mu_z \end{pmatrix} \tag{119}$$

where

$$\beta_1 = 1 + c_{jj} j + c_{ja} a + c_{jy} y + c_{jz} z \tag{120}$$

$$\beta_2 = 1 + c_{yj} j + c_{ya} a + c_{yy} y + c_{yz} z \tag{121}$$

We now study the stability of the 'axis', i.e. horizontal or vertical fixed points. First we consider the horizontal fixed points. So we consider the Jacobian matrix evaluated at $(j^*, a^*, 0, 0)$:

$$\begin{pmatrix} \frac{-b_1 c_{jj} a^*}{\beta_1^2} & \frac{b_1\beta_1 - b_1 a^* c_{ja}}{\beta_1^2} & \frac{-b_1 c_{jy} a^*}{\beta_1^2} & \frac{-b_1 c_{jz} a^*}{\beta_1^2} \\ 1-\mu_j & 1-\mu_a & 0 & 0 \\ 0 & 0 & 0 & \frac{b_2}{\beta^2} \\ 0 & 0 & 1-\mu_y & 1-\mu_z \end{pmatrix} \tag{122}$$

Because of the 2×2 block of zeros in the lower left corner, the multipliers of this 4×4 matrix are the multipliers of the 2×2 block in the upper left corner, and those of the 2×2 block in the lower right corner. The multipliers of the upper left block determine whether or not the horizontal fixed point is stable in the absence of competition by the second species. The

coefficients related to the second species or to the competition between the two species do not appear in the entries of this block. The multipliers of the lower right block determine if a fixed point that is stable within the axis will remain stable in the presence of an invading small number of the second species.
The characteristic polynomial of the lower right block is

$$\lambda^2 - (1-\mu_z)\lambda - (1-\mu_y)\frac{b_2}{\beta_2} \tag{123}$$

We first establish the conditions under which the roots λ_1 and λ_2 of (123) are inside the unit circle. By (111), it is necessary and sufficient that:

$$\frac{(1-\mu_y)b_2}{\mu_z\beta_2} < 1 \tag{124}$$

By substituting the value of β_2 in (121) into (124), and evaluating at $(j^*, a^*, 0, 0)$, we have :

$$\frac{b_2(1-\mu_y)}{\mu_z(1+c_{yj}j^* + c_{ya}a^*)} < 1 \tag{125}$$

So if the multipliers of the upper left 2 by 2 block in (122) are inside the unit circle, then (125) is a necessary and sufficient conditon for the stability of the horizontal fixed points.
A similar analysis shows that a vertical axis fixed point is stable if

$$\frac{b_1(1-\mu_j)}{\mu_a(1+c_{jy}y^* + c_{jz}z^*)} < 1 \tag{126}$$

We now consider the coexistence fixed points that is defined by a system of two linear equations for j^* and y^*:

$$\begin{array}{l} j^*(c_{jj} + c_{ja}\frac{1-\mu_j}{\mu_a}) + y^*(c_{jy} + c_{jz}\frac{1-\mu_y}{\mu_z}) = b_1\frac{1-\mu_j}{\mu_a} - 1 \\ j^*(c_{yj} + c_{ya}\frac{1-\mu_j}{\mu_a}) + y^*(c_{yy} + c_{yz}\frac{1-\mu_y}{\mu_z}) = b_2\frac{1-\mu_y}{\mu_z} - 1 \end{array} \tag{127}$$

The unique coexistence fixed point (j^*, a^*, y^*, z^*) is given by:

$$\begin{aligned} j^* &= \frac{\gamma(b_2\beta - 1) - (b_1\alpha - 1)\eta}{\delta\gamma - \epsilon\eta}, \\ a^* &= \alpha\left(\frac{\gamma(b_2\beta - 1) - (b_1\alpha - 1)\eta}{\delta\gamma - \epsilon\eta}\right), \\ y^* &= \frac{-\epsilon(b_2\beta - 1) + (b_1\alpha - 1)\delta}{\delta\gamma - \epsilon\eta}, \\ z^* &= \beta\left(\frac{-\epsilon(b_2\beta - 1) + (b_1\alpha - 1)\delta}{\delta\gamma - \epsilon\eta}\right), \end{aligned}$$

provided

$$H \equiv \delta\gamma - \epsilon\eta \neq 0, \tag{128}$$

where

$$\alpha = \frac{1-\mu_j}{\mu_a}, \quad \beta = \frac{1-\mu_y}{\mu_z}, \quad \epsilon = c_{jj} + c_{ja}\alpha$$

and

$$\gamma = c_{jy} + c_{jz}\beta, \quad \delta = c_{yj} + c_{ya}\alpha, \quad \eta = c_{yy} + c_{yz}\beta.$$

The resulting eigenvalue equation of the Jacobian matrix evaluated in (j^*, a^*, y^*, z^*) is quartic, but we can compute the multipliers numerically in MatContM if the actual values of state variables and parameters are known. In this way we will be able to determine the stability of fixed points numerically.
We will study the overall dynamics of the model for the following fixed model parameter values which we took from [58] .

$b_1 = 20$	$c_{jj} = 0.36$	$b_2 = 18$	$c_{ja} = 0.55$	$c_{jz} = 0.23$	$\mu_j = 0.23$
$\mu_a = 0.72$	$c_{ya} = 0.08$	$c_{yy} = 0.18$	$c_{yz} = 0.26$	$\mu_y = 0.29$	$\mu_z = 0.98$

The parameters c_{jy} and c_{yj} will be used as free parameters.

10.2.2 Numerical continuation of the horizontal fixed points and their stability analysis

First we consider horizontal fixed points and their continuation. For all values of the parameters c_{jy} and c_{yj} , the fixed point obtained from (113) and (114)

$$F_H = (21.50285631, 22.99611022, 0, 0)$$

remains unchanged since c_{jy} and c_{yj} do not appear in (113) and (114). Also, the free parameters do not enter in the the entries of the left upper block 2 by 2 matrix of (122). We first show that all the multipliers of this block are inside the unit circle. The 2 by 2 upper left block is:

$$\begin{pmatrix} \frac{-b_1 c_{jj} a^*}{\beta_1^2} & \frac{b_1\beta_1 - b_1 a^* c_{ja}}{\beta_1^2} \\ 1-\mu_j & 1-\mu_a \end{pmatrix} \tag{129}$$

The multipliers of (129) in F_H are -0.6712 and 0.5893, with absolute values less than 1. That means that in a continuation of the horizontal fixed points with either c_{yj} or c_{jy} free, the multipliers of the upper left block of (122) are inside the unit circle. Now we consider the stability condition (125). For the given model parameters, the horizontal fixed point is stable if :

$$\frac{12.78}{21.0728 \times c_{yj} + 2.7829} < 1 \tag{130}$$

This is equivalent to $c_{yj} > c_{yj0}$ where $c_{yj0} = 0.474477674$. Hence F_H is unstable if $0 \le c_{yj} < c_{yj0}$ and stable if $c_{yj0} < c_{yj} \le 1$. It is biologically plausible that the horizontal fixed point is stable only if the juveniles of the first species suppress the juveniles of the second species to a sufficient degree. Now we perform the continuation, where in $mapfile$ the order of the state variables and parameters is (j, a, y, z) and $(b_1, b_2, \mu_a, \mu_j, \mu_y, \mu_z, c_{jj}, c_{ja}, c_{yy}, c_{yz}, c_{jy}, c_{jz}, c_{yj}, c_{ya})$, respectively. First we vary only the parameter c_{yj} from 0 to 1, while c_{jy} is fixed at 0; we refer to this as *Run 1*:

```
>> init
>> global  opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
```

```
>> ap=13; p=[20;18;0.72;0.23;0.29;0.98;0.36;0.55;0.18;0.26;0.;0.23;0.;.08];
>> opt = contset(opt,'MaxNumPoints',50);
>> opt = contset(opt,'Singularities',1);
>> [x0,v0]=init_FPm_FPm(@LeslieGower,[21.50285631;22.99611022;0;0], p, ap,1);
>> [x12,v12,s12,h12,f12]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = NS, x = ( 21.502856 22.996110 0.000000 0.000000 0.076818 )
Neutral saddle
label = NS, x = ( 21.502856 22.996110 0.000000 0.000000 0.132182 )
Neutral saddle
label = PD, x = ( 21.502856 22.996110 0.000000 0.000000 0.450625 )
normal form coefficient of PD = 3.260623e-003
label = BP, x = ( 21.502856 22.996110 0.000000 0.000000 0.474408 )
elapsed time  = 2.1 secs
npoints curve = 50
```

This test is run by typing *LeslieGower12* in the command window.
By monitoring the multipliers in *Run 1*, we also find that the horizontal fixed point is unstable before the BP point and stable afterwards.
To understand the appearance of the PD point in *Run 1*, we see that F_H is a PD point if there is an eigenvalue -1, which is possible only if:

$$\frac{1}{2}[(1-\mu_z) - \sqrt{(1-\mu_z)^2 + 4(1-\mu_y)\frac{b_2}{\beta_2}}] = -1 \tag{131}$$

From this equation, β_2 and hence c_{yj} can be computed. For the model parameters, this implies that $c_{yj} = 0.4506$, independently of c_{jy}. This is the same value as found in *Run 1*.
In *Run 1* we also detected a BP point. In this point the characteristic polynomial (123) must have a root $+1$, i.e.

$$1 - (1-\mu_z) - (1-\mu_y)\frac{b_2}{\beta_2} = 0 \tag{132}$$

If we input the fixed values of the model parameters, then the value of parameter c_{yj} that satisfies (132) is precisely c_{yj0}; this is also the value found for the BP point in *Run 1*. We now check that for this value of the continuation parameter the Jacobian $[(M_{LG})_x - I|(M_{LG})_{c_{yj}}]$ is rank deficient. The Jacobian evaluated in a horizontal fixed point is:

$$[(M_{LG})_x - I|\ (M_{LG})_{c_{yj}}] = \begin{pmatrix} \frac{-b_1 c_{jj} a^*}{\beta_1^2} - 1 & \frac{b_1\beta_1 - b_1 a^* c_{ja}}{\beta_1^2} & \frac{-b_1 c_{jy} a^*}{\beta_1^2} & \frac{-b_1 c_{jz} a^*}{\beta_1^2} & 0 \\ 1-\mu_j & -\mu_a & 0 & 0 & 0 \\ 0 & 0 & -1 & \frac{b_2}{\beta^2} & 0 \\ 0 & 0 & 1-\mu_y & -\mu_z & 0 \end{pmatrix} \tag{133}$$

From this form it is clear that if the lower right two by two block of $[f_x - I]$ is singular in a horizontal fixed point, then $[(M_{LG})_x - I|\ (M_{LG})_{c_{yj}}]$ is rank deficient, i.e. we have a BP point. This confirms the result of the continuation concerning the BP point. Now we switch to a new branch in the BP point of *Run 1*. This is *Run 2*:

```
>> global x12 v12 s12  opt cds  fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt=contset(opt,'Singularities',1);
>> opt = contset(opt,'MaxNumPoints',250);
>> x1=x12(1:4,s12(5).index);p1=p;
>> p1(fpmds.ActiveParams)=x12(5,s12(5).index);
>> [x2,v2]=init_BPm_FPm(@LeslieGower,x1,p1,s12(5),0.01);
>> [x131,v131,s131,h131,f131]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = PD, x = ( 21.984277 23.510963 -2.739442 -1.984697 0.508048 )
normal form coefficient of PD = 2.453609e-003

elapsed time  = 1.3 secs
npoints curve = 250
>> opt = contset(opt,'Backward',1);
>> [x132,v132,s132,h132,f132]=cont(@fixedpointmap,x2,[],opt);
first point found
tangent vector to first point found
label = BP, x = ( 21.502856 22.996110 -0.000000 -0.000000 0.474408 )

elapsed time  = 1.5 secs
npoints curve = 250
```

This test is run by typing *LeslieGower13* in the command window.
The branch with negative components that we found in the *forward* continuation in *Run 2* is not biologically meaningful. By changing the direction of continuation we detected the same `BP` point as in *Run 1*.
We also continue the fixed points of M_{LG}^2 starting from the `PD` point in *Run 1*. We call this *Run 3*:

```
>> global x12 v12 s12  opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt = contset(opt,'MaxNumPoints',100);
>> opt=contset(opt,'Singularities',1);
>> x1=x12(1:4,s12(4).index);p1=p;
>> p1(fpmds.ActiveParams)=x12(5,s12(4).index);
>> [x5,v5]=init_PDm_FP2m(@LeslieGower,x1,p1,s12(4),0.01,1);
>> opt=contset(opt,'Backward',1);
>> [x14,v14,s14,h14,f14]=cont(@fixedpointnmap,x5,v5,opt);
first point found
tangent vector to first point found
label = BP, x = ( 21.502856 22.996110 0.000000 -0.000000 0.450625 )
label = NS, x = ( 19.656619 23.483812 6.223217 -0.928635 0.402732 )
Neutral saddle
elapsed time  = 1.1 secs
```

```
npoints curve = 100
```

This test is run by typing *LeslieGower14* in the command window.
The BP point in *Run 3* is the PD point in *Run 1* . By monitoring the multipliers of M_{LG}^2 we find that the fixed point of M_{LG}^2 is unstable. In fact in the PD point

$$x = (21.5028, 22.9961, 0.0000, 0.0000, 0.450625)$$

the multipliers of M_{LG} and M_{LG}^2 are
$(0.5893, -0.6712, -1.0000, 1.0200)$ and $(0.3473, 0.4506, 1.0404, 1.0000)$ respectively.
In the nearby point

$$x = (21.7097, 22.8452, -0.4689, 0.4632, 0.4487)$$

the multipliers of M_{LG} and M_{LG}^2 are
$(-0.6742, 0.9928, 0.5913, 1.0099)$ and $(0.3473, 0.4516, 0.9938, 1.0485)$ respectively.

By (126), the vertical fixed point is stable for the fixed values of the model parameters if :

$$\frac{15.4}{23.5346c_{jy} + 4.6416} < 1 \tag{134}$$

That is equivalent to $c_{jy} > c_{jy0}$ where $c_{jy0} = 0.4571312026$. So the vertical fixed point is unstable for $0 \leq c_{jy} < c_{jy0}$ and stable for $c_{jy0} < c_{jy} \leq 1$. It is biologically plausible that the vertical fixed point is stable only if the juveniles of the second species suppress the juveniles of the first species to a sufficient degree.

10.2.3 Numerical continuation of the vertical fixed points and their stability analysis

We now start the continuation from the vertical fixed point $(0, 0, 32.68698060, 23.68138391)^T$ where $c_{yj} = c_{jy} = 0$ (116) and (117). We vary the parameter c_{jy} from 0 to 1 while c_{yj} is held fixed at 0. This is called *Run 4*.

```
>> global opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt=contset(opt,'MaxNumPoints',200);
>> opt=contset(opt,'Singularities',1);
>> ap=11; p=[20;18;0.72;0.23;0.29;0.98;0.36;0.55;0.18;0.26;0;0.23;0;.08];
>> x=[0;0;32.68698060;23.68138391];
>> [x0,v0]=init_FPm_FPm(@LeslieGower,x, p, ap,1);
>> [x15,v15,s15,h15,f15]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = NS, x = ( 0.000000 0.000000 32.686981 23.681384 0.152939 )
Neutral saddle
label = PD, x = ( 0.000000 0.000000 32.686981 23.681384 0.170849 )
normal form coefficient of PD = 1.304652e-005
```

```
label = BP, x = ( 0.000000 0.000000 32.686981 23.681384 0.457129 )
elapsed time  = 1.6 secs
npoints curve = 200
```

This test is run by typing *LeslieGower15* in the command window.
By monitoring the multipliers in *Run 4* we find that the vertical fixed point is unstable before the BP point and stable afterwards; the parameter value c_{jy0} predicted for the BP point is indeed found in *Run 4*. If we vary the parameter c_{yj} while c_{jy} is held fixed at 0, then by monitoring the multipliers we see that the vertical fixed point is unstable for all values of c_{yj}. It is not hard to explain this. Since in the stability condition (126), the parameter c_{yj} is absent and the left hand side of (126) has the value 3.3178 for the model parameter values, and $c_{jy} = 0$, we conclude that the vertical fixed point is unstable on the c_{yj} axis.

Now we continue the fixed points of M_{LG}^2 starting from the PD point in *Run 4*. We call this *Run 5*:

```
>> global x15 v15 s15  opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> x1=x15(1:4,s15(3).index);p1=p;
>> p1(fpmds.ActiveParams)=x15(5,s15(3).index);
>> opt=contset(opt,'Singularities',1);
>> [x2,v2]=init_PDm_FP2m(@LeslieGower,x1,p1,s15(3),0.01,1);
>> [x16,v16,s16,h16,f16]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = LP  , x = ( -0.002692 0.001619 32.694773 23.675960 0.170849 )
normal form coefficient of LP =-7.827823e-07
label = LP  , x = ( -11.313105 6.463612 54.415142 9.852869 0.167365 )
normal form coefficient of LP =-1.750138e-03
elapsed time  = 1.2 secs
npoints curve = 300
```

We now switch to a new branch in the BP point of *Run 4*. This is *Run 6*:

```
>> global x15 v15 s15 opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt = contset(opt,'MaxNumPoints',200);
>> opt=contset(opt,'Backward',1);
>> opt=contset(opt,'Singularities',1);
>> x1=x15(1:4,s15(4).index);p1=p;p1(fpmds.ActiveParams)=x15(5,s15(4).index);
>> [x2,v2]=init_BPm_FPm(@LeslieGower,x1,p1,s15(4),0.01,1);
>> [x17,v17,s17,h17,f17]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = BP, x = ( 0.000000 0.000000 32.686981 23.681384 0.457129 )
elapsed time  = 1.2 secs
npoints curve = 200
```

This test is run by typing *LeslieGower17* in the command window.
The BP point is the same as in *Run 4* and the new branch is a branch of coexistence fixed points.

10.2.4 Numerical continuation of the coexistence fixed points and their stability analysis

Now we consider the coexistence fixed points (j^*, a^*, y^*, z^*), starting the continuation from

$$(16.42912, 17.570032, 28.871217, 20.916902)$$

where $c_{yj} = c_{jy} = 0$. This fixed point bifurcates to vertical and horizontal fixed points respectively, when one of c_{jy} and c_{yj} is varied and the other variable is held fixed at 0. In the model this means that one species drives another to extinction. We first do continuation of coexistence fixed points, where c_{jy} is the free parameter. This is *Run 7*:

```
>> global opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt=contset(opt,'MaxNumPoints',350);
>> opt=contset(opt,'Singularities',1);
>> opt=contset(opt,'Backward',1);
>> ap=11; p=[20;18;0.72;0.23;0.29;0.98;0.36;0.55;0.18;0.26;0;0.23;0;.08];
>> x=[16.42912;17.570032;28.871217;20.916902];
>> [x0,v0]=init_FPm_FPm(@LeslieGower,x, p, ap,1);
>> [x18,v18,s18,h18,f18]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = BP, x = ( 0.000000 0.000000 32.686981 23.681384 0.457129 )
elapsed time  = 1.8 secs
npoints curve = 350
>>
```

This test is run by typing *LeslieGower18* in the command window.
In *Run 7* we see that the coexistence fixed points bifurcate to vertical fixed points in the BP point where $c_{jy} = c_{jy0}$. For parameter values $c_{jy} > c_{jy0}$ the coexistence fixed point is not biologically meaningful because its first and second components become negative. The coexistence fixed point is stable before the BP point and unstable afterwards.
Now we switch to a new branch in the BP point of *Run 7*. This is *Run 8*:

```
>> global x18 v18 s18  opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt = contset(opt,'MaxNumPoints',50);
>> opt = contset(opt,'Singularities',1);
>> opt = contset(opt,'Backward',1);
>> x1=x18(1:4,s18(2).index);p1=p;p1(fpmds.ActiveParams)=x18(5,s18(2).index);
>> [x2,v2]=init_BPm_FPm(@LeslieGower,x1,p1,s18(2),0.01);
>> [x19,v19,s19,h19,f19]=cont(@fixedpointmap,x2,v2,opt);
```

```
first point found
tangent vector to first point found
label = BP, x = ( 0.000000 0.000000 32.686981 23.681384 0.457129 )
label = PD, x = ( 0.000000 0.000000 32.686981 23.681384 0.170849 )
normal form coefficient of PD = 1.304649e-005
label = NS, x = ( 0.000000 0.000000 32.686981 23.681384 0.152939 )
Neutral saddle
elapsed time  = 0.5 secs
npoints curve = 12
```

This test is run by typing *LeslieGower19* in the command window.
The BP point is the same as in *Run 4* and the new branch is a branch of vertical fixed points and contains a PD point.

In the next *Run 9* we continue the coexistence fixed points with free parameter c_{yj}:

```
>> global opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt=contset(opt,'MaxNumPoints',400);
>> opt=contset(opt,'Singularities',1);
>> ap=13; p=[20;18;0.72;0.23;0.29;0.98;0.36;0.55;0.18;0.26;0;0.23;0;.08];
>> x=[16.42912;17.570032;28.871217;20.916902];
>> [x0,v0]=init_FPm_FPm(@LeslieGowere,x, p, ap,1);
>> [x20,v20,s20,h20,f20]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = BP, x = ( 21.502856 22.996110 0.000000 0.000000 0.474408 )
elapsed time  = 2.0 secs
npoints curve = 400
```

This test is run by typing *LeslieGower20* in the command window.
We see that the coexistence fixed points bifurcate to horizontal fixed points in the BP point where $c_{yj} = c_{yj0}$. Therefore for the parameter values $c_{yj} > c_{yj0}$ the coexistence fixed point is not biologically meaningful due to the first and the second components of it being negative. The coexistence fixed point is stable before the BP point and unstable afterwards.
We now switch to a new branch in the BP point of *Run 9*, and refer to this as *Run 10*:

```
>> global x20 v20 s20 opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> x1=x20(1:4,s20(2).index);p1=p;
>> p1(fpmds.ActiveParams)=x20(5,s20(2).index);
>> opt = contset(opt,'MaxNumPoints',50);
>> opt = contset(opt,'Backward',1);
>> [x2,v2]=init_BPm_FPm(@LeslieGower,x1,p1,s20(2),0.01,1);
>> [x21,v21,s21,h21,f21]=cont(@fixedpointmap,x2,v2,opt);
first point found
```

```
tangent vector to first point found
label = BP, x = ( 21.502856 22.996110 0.000000 0.000000 0.474408 )
label = PD, x = ( 21.502856 22.996110 0.000000 0.000000 0.450625 )
normal form coefficient of PD = 3.260622e-003
elapsed time  = 0.8 secs
npoints curve = 50
```

This test is run by typing *LeslieGower21* in the command window.
The BP point in *Run 10* is the same BP point as in *Run 9*, the points computed in *Run 10* are horizontal fixed points.
The solutions to the equation $H = 0$, where H is given by (128), are the parameter values for which the existence and uniqueness of the coexistence fixed point is not guaranteed. In the present context, where only c_{yj} and c_{jy} are not fixed, this leads to

$$H \equiv c_{yj}c_{jy} + ac_{yj} + bc_{jy} - c = 0, \tag{135}$$

where

$$a = 0.1666326531, b = 0.0855555552, c = 0.3350275226.$$

The equation $H = 0$ defines a hyperbola in (c_{yj}, c_{jy}) space.

Numerically we also find that the special point (c_{yj0}, c_{jy0}) where both the horizontal and vertical fixed point lose their stability lies on the hyperbola. To check that this is not an accident we note that by (125) we have

$$c_{yj0} = \frac{b_2(1-\mu_y) - \mu_z(1 + c_{ya}a^*)}{\mu_z j^*} \tag{136}$$

where j^* and a^* are given by (113) and (114). Similarly by (126)

$$c_{jy0} = \frac{b_1(1-\mu_j) - \mu_a(1 + c_{jz}z^*)}{\mu_a y^*} \tag{137}$$

where y^* and z^* are given by (116) and (117). By substituting (136) and (137) in H and simplifying the result in Maple, we see that:

$$\begin{array}{c} c_{yj0}c_{jy0} + c_{jy0}c_{ya}(\frac{1-\mu_j}{\mu_a}) + c_{jz}c_{yj0}(\frac{1-\mu_y}{\mu_z}) + c_{jz}c_{ya}(\frac{1-\mu_j}{\mu_a}(\frac{1-\mu_y}{\mu_z}) \\ -(c_{yj} + c_{ya}\frac{1-\mu_j}{\mu_a})(c_{jy} + c_{jz}\frac{1-\mu_y}{\mu_z}) = 0 \end{array} \tag{138}$$

That means that the point (c_{yj0}, c_{jy0}) is on the hyperbola.
To study the solutions of the system of equations, of the coexistence fixed points, for the parameter values where $H = 0$, we define

$$E_1 \equiv \det \begin{pmatrix} \epsilon & b_1\alpha - 1 \\ \delta & b_2\beta - 1 \end{pmatrix} = 0 \tag{139}$$

and

$$E_2 \equiv \det \begin{pmatrix} \gamma & b_1\alpha - 1 \\ \eta & b_2\beta - 1 \end{pmatrix} = 0 \tag{140}$$

(139) and (140) are linear equations in c_{yj} and c_{jy} respectively that are satisfied iff $c_{yj} = c_{yj0}$ and $c_{jy} = c_{jy0}$ respectively.

This means that only for the parameters values $(c_{yj}, c_{jy}) = (c_{yj0}, c_{jy0})$ a coexistence fixed point can lie on the hyperbola $H = 0$.

The above argument also shows that in (127) the right hand side vector is in the column space of the coefficient matrix, i.e. there is a straight line of coexistence fixed points for the parameter values $(c_{yj}, c_{jy}) = (c_{yj0}, c_{jy0})$.

We notice that the equation

$$j^*(c_{yj} + c_{ya} \times \alpha) = b_2\beta - 1 \tag{141}$$

is satisfied where $c_{yj} = c_{yj0}$ and j^* is a component of the horizontal fixed point given by (113). Similarly, the equation

$$y^*(c_{jy} + c_{jz} \times \beta) = b_1\alpha - 1 \tag{142}$$

is satisfied where $c_{jy} = c_{jy0}$ and y^* is a component of the vertical fixed point given by (116). The above arguments show that the line of coexistence fixed points bifurcates to the horizontal and vertical fixed points when $c_{jy} = c_{jy0}$ and $c_{yj} = c_{yj0}$ respectively. This reconfirms the results obtained in *Run 7* and *Run 9*, where we found the new branches of fixed points in c_{jy0} and c_{yj0} respectively.

The straight line of coexistence fixed points contains a horizontal fixed point, a vertical fixed point and two coexistence flip points. As a curve, it is a curve of fold bifurcations of fixed points. This explains why the coexistence fixed points on the curve of *Run 16* are detected as fold-flip points.

Now we discuss the stability of the coexistence fixed points. The coexistence fixed point is unstable in S_2 where

$$S_2 = \{(c_{yj}, c_{jy}) : c_{yj} > c_{jy0} \quad or \quad c_{jy} > c_{jy0}\}$$

The stability properties of the coexistence fixed point are more complicated in S_1 where

$$S_1 = \{(c_{yj}, c_{jy}) : 0 \le c_{yj} \le c_{yj0}, \quad 0 \le c_{jy} \le c_{jy0}\}.$$

A picture of the hyperbola $H = 0$, the rectangle S_1 and the region S_2 are given in Figure 12. By numerical continuation runs in this rectangle it is found that there is an interior region in which the coexistence fixed points are unstable. This region is bounded by a curve of `PD` points where the stability changes. We now give some detailed computations. We first consider *Run 11* which starts from the coexistence fixed point

$$(11.17052012, 11.94625068, 20.99523834, 15.21083594)$$

where $c_{yj} = 0.3$ and $c_{jy} = 0.3$; c_{jy} is free in the continuation:

```
>> global opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt=contset(opt,'MaxNumPoints',900);
>> opt=contset(opt,'Singularities',1);
```

```
>> ap=11; p=[20;18;0.72;0.23;0.29;0.98;0.36;0.55;0.18;0.26;0.3;0.23;0.3;.08];
>> x=[11.17052012;11.94625068;20.99523834;15.21083594];
>> [x0,v0]=init_FPm_FPm(@LeslieGower,x, p, ap,1);
>> [x221,v221,s221,h221,f221]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = PD, x = ( 13.759070 14.714561 18.285905 13.247952 0.234912 )
normal form coefficient of PD = 4.356836e-005
elapsed time  = 3.5 secs
npoints curve = 900
>> opt=contset(opt,'Backward',1);
>> [x222,v222,s222,h222,f222]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = PD, x = ( 2.420127 2.588191 30.153929 21.846214 0.433426 )
normal form coefficient of PD = -1.146987e-004
label = BP, x = ( -0.000000 -0.000000 32.686981 23.681384 0.457129 )
elapsed time  = 3.7 secs
npoints curve = 900
```

This test is run by typing *LeslieGower22* in the command window.
In *Run 11* the coexistence fixed point is stable when $c_{jy} < 0.234912$ (the first PD point) , and again stable between the second PD point and the BP point. For other parameter values, in particular between the two PD points, it is unstable.

We now do a continuation with c_{yj} free, and call this *Run 12*:

```
>>>> global opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt=contset(opt,'MaxNumPoints',900);
>> opt=contset(opt,'Singularities',1);
>> ap=13; p=[20;18;0.72;0.23;0.29;0.98;0.36;0.55;0.18;0.26;0.3;0.23;0.3;.08];
>> x=[11.17052012;11.94625068;20.99523834;15.21083594];
>> [x0,v0]=init_FPm_FPm(@LeslieGower,x, p, ap,1);
>> [x231,v231,s231,h231,f231]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = PD, x = ( 19.493527 20.847244 4.082945 2.958052 0.454972 )
normal form coefficient of PD = -2.843076e-005
label = BP, x = ( 21.502856 22.996110 -0.000000 -0.000000 0.474408 )
elapsed time  = 3.8 secs
npoints curve = 900
>> opt=contset(opt,'Backward',1);
>> [x232,v232,s232,h232,f232]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = PD, x = ( 8.793401 9.404053 25.825530 18.710333 0.201880 )
normal form coefficient of PD = 2.307707e-005
```

```
elapsed time  = 3.5 secs
npoints curve = 900
```

This test is run by typing *LeslieGower23* in the command window.
The coexistence fixed point is stable for $c_{yj} < 0.201880$ (the first PD point) and between the second PD point and the BP. For other parameter values, in particular between the two PD points, it is unstable.
We now do a continuation of the fixed points of M_{LG}^2, starting in the PD point in *Run 11* which corresponds to $c_{yj} = 0.234912$, we call this *Run 13*:

```
>> global x221 v221 s221  opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt = contset(opt,'Backward',1);
>> opt = contset(opt,'Singularities',1);
>> x1=x221(1:4,s221(2).index);p1=p;
>> p1(fpmds.ActiveParams)=x221(5,s221(2).index);
>> [x2,v2]=init_PDm_FP2m(@LeslieGower,x1,p1,s221(2),0.01,1);
>> [x24,v24,s24,h24,f24]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = BP, x = ( 13.759070 14.714561 18.285905 13.247952 0.234912 )
label = LP, x = ( 2.198620 6.212272 10.294425 33.964752 0.504965 )
normal form coefficient of LP =-4.246467e-003
label = BP, x = ( 2.420127 2.588191 30.153929 21.846214 0.433426 )
label = LP, x = ( 6.819750 3.432374 47.612656 7.988337 0.504965 )
normal form coefficient of LP =-4.625801e-003
Closed curve detected at step 1183
elapsed time  = 6.6 secs
npoints curve = 1183
>> cpl(x24,v24,s24,[1 5])
```

This test is run by typing *LeslieGower24* in the command window.
The continuation leads to a closed curve of fixed points of M_{LG}^2. A picture of this closed curve is given in Figure 10. The fixed points are initially stable, then unstable between two successive LP points, then stable again. The multipliers of M_{LG}^2 at the begining, just before the LP points, just after the LP points and near the end of the closed curve are given by v_1, v_2, v_3, v_4, v_5 and v_6 respectively:
$v_1 = (1.0000, 0.8263, 0.4771, 0.3573)$, $v_2 = (0.2905, 0.3609, 0.9997, 0.8401)$
$v_3 = (1.0207, 0.8731, 0.4057, 0.2844)$, $v_4 = (0.2892, 0.3640, 0.8420, 1.0017)$
$v_5 = (0.2930, 0.3559, 0.8372, 0.9964)$, $v_6 = (0.3574, 0.4769, 0.9998, 0.8263)$

Now we do a continuation of coexistence fixed points starting from

$$(21.49714644, 22.99000382, 0.03249100281, 0.02353939999)$$

where $c_{yj} = 0.474$ and $c_{jy} = 0$, the free parameter is c_{jy}, and call this *Run 14*:

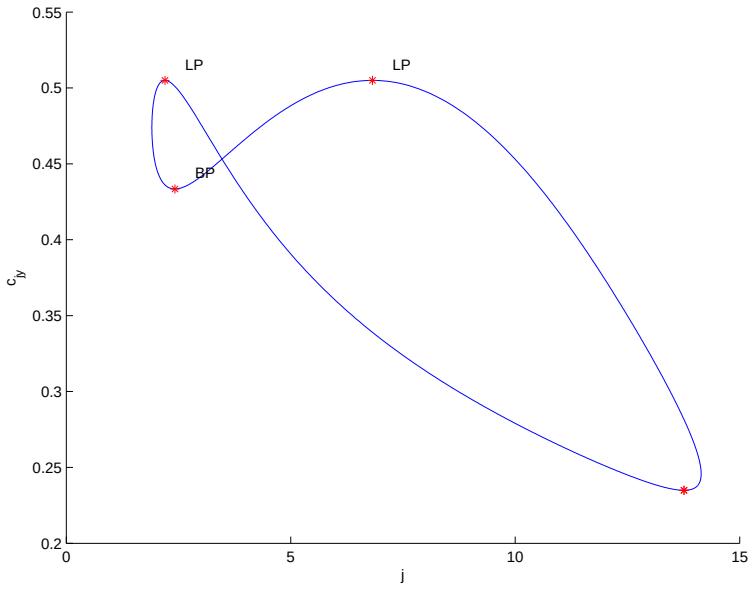


Figure 10: The second iterate closed curve in (j, c_{jy}) space.

```
>> global opt cds fpmds
>> opt = contset;
>> ap=11;p=[20;18;0.72;0.23;0.29;0.98;0.36;0.55;0.18;0.26;0;0.23;0.474;.08];
>> x=[21.49714644;22.99000382;0.03249100281;0.02353939999];
>> [x0,v0]=init_FPm_FPm(@LeslieGower,x, p, ap,1);
>> opt=contset(opt,'MaxNumPoints',1200);
>> opt=contset(opt,'Singularities',1);
>> opt=contset(opt,'Backward',1);
>> opt = contset(opt,'Multipliers',1);
>> [x25,v25,s25,h25,f25]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = PD, x = ( 20.336353 21.748600 1.795753 1.301005 0.449305 )
normal form coefficient of PD = -1.491141e-001
label = PD, x = ( 1.299106 1.389322 30.713618 22.251703 0.457100 )
normal form coefficient of PD = -1.448721e-001
label = BP, x = ( -0.000000 -0.000000 32.686981 23.681384 0.457129 )
elapsed time  = 5.2 secs
npoints curve = 1200
```

This test is run by typing *LeslieGower25* in the command window.
Now we do a continuation of fixed points of M_{LG}^2 by starting in the `PD` point of *Run 14* that corresponds to $c_{jy} = 0.449305$, we call this *Run 15*:

```
> global x25 v25 s25 h25 f25 opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
```

```
>> x1=x25(1:4,s25(2).index);p1=p;
>> p1(fpmds.ActiveParams)=x25(5,s25(2).index);
>> opt = contset(opt,'MaxNumPoints',1700);
>> opt=contset(opt,'Singularities',1);
>> opt=contset(opt,'Backward',0);
>> [x2,v2]=init_PDm_FP2m(@LeslieGower,x1,p1,s25(2),0.01,1);
>> [x26,v26,s26,h26,f26]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = LP, x = ( 14.881412 19.584062 2.063890 10.875613 0.335687 )
normal form coefficient of LP =1.853991e-003
label = LP, x = ( 1.211432 6.467511 5.173194 37.643568 0.739309 )
normal form coefficient of LP =1.007065e-002
label = BP, x = ( 1.299106 1.389322 30.713618 22.251703 0.457100 )
label = LP, x = ( 7.401653 2.743706 52.894438 4.425839 0.739309 )
normal form coefficient of LP =9.716217e-003
label = LP, x = ( 19.273038 16.942225 15.270360 1.682874 0.335687 )
normal form coefficient of LP =-1.436619e-003
Closed curve detected at step 1665
elapsed time  = 9.4 secs
npoints curve = 1665
```

This test is run by typing *LeslieGower26* in the command window.
In the continuation there is a pair of LP points for $c_{jy} = 0.335687$, and another pair of LP points for $c_{jy} = 0.739309$. The fixed points of M_{LG}^2 are stable only between two successive LP points.

For a detailed study of the behavior of coexistence fixed points in S_2, we compute some bifurcation curves. First we compute the period doubling curve that originates in the PD point of *Run 11*, call this *Run 16*:

```
>> global  x221 v221 s221 opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt=contset(opt,'Maxnumpoints',250);
>> opt=contset(opt,'Singularities',1);
>> ap1=11; p1=p;
>> p1(ap1)=x221(end,s221(2).index);
>> x1=x221(1:4,s221(2).index);
>> [x0,v0]=init_PDm_PDm(@LeslieGower,x1,p1,[11 13],1);
>> [x27,v27,s27,h27,f27]=cont(@perioddoublingmap,x0,v0,opt);
first point found
tangent vector to first point found
label = GPD , x = ( 19.463734 20.815382 4.163299 3.016267 0.297779 0.454279 )
Normal form coefficient of GPD = 5.271115e-006
label = LPPD, x = ( 20.354989 21.768530 1.744899 1.264162 0.457129 0.474408 )
```

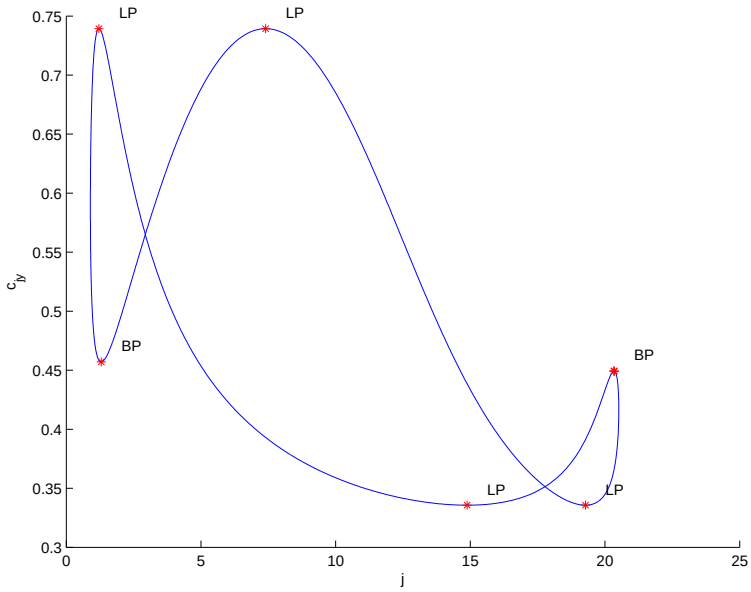


Figure 11: A closed curve of fixed points of M_{LG}^2 in (j, c_{jy}) space.

```
Normal form coefficient for LPPD :[a/e , be]= -1.329134e-009, -5.060725e-005,

elapsed time  = 4.2 secs
npoints curve = 250
>> opt=contset(opt,' Backward',1);
>> opt=contset(opt,'MaxNumPoints',270);
>>  [x1,v1,s1,h1,f1]=cont(@perioddoublingmap,x0,v0,opt);
first point found
tangent vector to first point found
label = GPD , x = ( 4.771088 5.102414 28.857159 20.906717 0.383143 0.210138 )
Normal form coefficient of GPD = 4.494110e-008
label = LPPD, x = ( 1.297714 1.387833 30.714296 22.252194 0.457129 0.474408 )
Normal form coefficient for LPPD :[a/e , be]= -1.076502e-008, -2.619423e-006,

elapsed time  = 4.1 secs
npoints curve = 270
```

This test is run by typing *LeslieGower27* in the command window.
We can explain analytically the detection of the $LPPD$ point in *Run 16*. As shown before, for the parameters value $(c_{yj}, c_{jy}) = (c_{yj0}, c_{jy0})$ there is a curve of fixed points. This curve can be defined by $X = X(t)$ where t is arclength, with constant parameter $\alpha = \alpha_0$. The fixed point equation is given by

$$f(X(t), \alpha_0) - X(t) = 0 \tag{143}$$

By taking derivatives of (143) with respect to t, we have

$$f_X(X(t), \alpha_0)\frac{dX}{dt} - \frac{dX}{dt} = (f_X(X(t), \alpha_0) - I)\frac{dX}{dt} = 0. \tag{144}$$

Here $\frac{dX}{dt} \neq 0$ since $\frac{dX}{dt}$ is a unit vector tangent to the curve of fixed points where $\alpha = \alpha_0$. We conclude that in (144), $(f_X(X(t), \alpha_0) - I)$ must be rank deficient , i.e, each point is a fold point. This argument explains the detection of $LPPD$ points in *Run 16*. Now we compute the limit point curve of M^2_{LG}, starting from the `LP` point of M^2_{LG} in *Run 13*. We call this *Run 17*:

```
>>>>>>>>>>>>>>fold curve <<<<<<<<<<<
>> p1(fpmds.ActiveParams)=x24(end,s24(4).index);
>>x1=x24(1:4,s24(4).index);
>> opt=contset(opt,'Maxnumpoints',120);
>> [x0,v0]=init_LPm_LPm(@LeslieGower,x1,p1,[11 13],2);
>>opt=contset(opt,'Backward',0);
>> [x1,v1,s1,h1,f1]=cont(@limitpointmap,x0,v0,opt);
first point found
tangent vector to first point found
elapsed time  = 2.2 secs
npoints curve = 120
>> cpl(x1,v1,s1,[6 5]);
>> opt=contset(opt,'Backward',1);
>>[x2,v2,s2,h2,f2]=cont(@limitpointmap,x0,v0,opt);
first point found
tangent vector to first point found
elapsed time  = 2.3 secs
npoints curve = 120
>> cpl(x2,v2,s2,[6 5]);
```

This test is run by typing *LeslieGower28* in the command window.
In the next run, *Run 18*, we continue the fixed points of M^2_{LG} starting from the `PD` point in *Run 13*, detect an `LP` point and then continue the fold curve of M^2_{LG}.

```
>>>>>>>>>>> fold curve <<<<<<<<<<
>>opt=contset(opt,'Backward',0);
>> x1=x24(1:4,s24(4).index);p1=p;
>> p1(fpmds.ActiveParams)=x24(5,s24(4).index);
>> opt=contset(opt,'MaxNumPoints',250);
>> [x2,v2]=init_LPm_LPm(@LeslieGower,x1,p1,[11 13],2);
>>[x3,v3,s3,h3,f3]=cont(@limitpointmap,x2,v2,opt);
first point found
tangent vector to first point found

elapsed time  = 4.7 secs
npoints curve = 250
>> cpl(x3,v3,s3,[6 5]),
>>opt=contset(opt,'Backward',1);
>> [x3,v3,s3,h3,f3]=cont(@limitpointmap,x2,v2,opt);
first point found
tangent vector to first point found
```

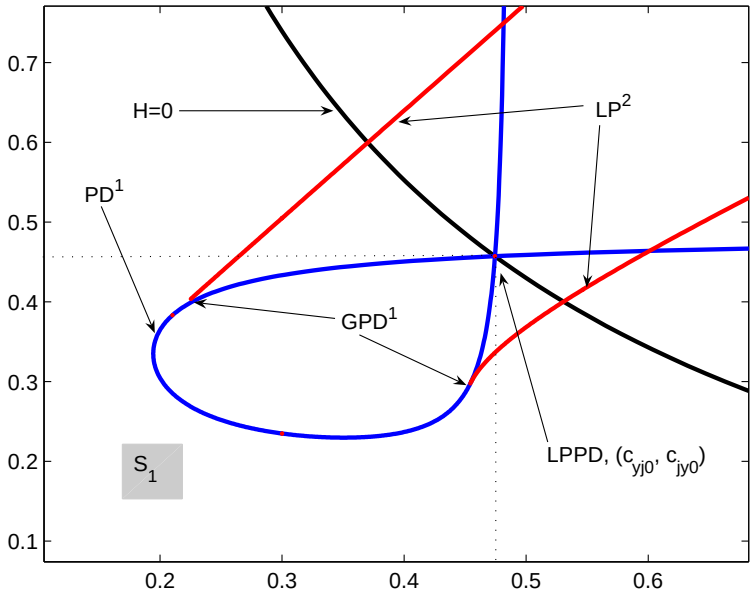


Figure 12: The flip curve, the fold curve of M^2_{LG}, the hyperbola $H = 0$ and the rectangle S_1 in (c_{yj}, c_{jy}) space.

```
elapsed time  = 4.2 secs
npoints curve = 250
>> cpl(x3,v3,s3,[6 5]);
```

This test is run by typing *LeslieGower29* in the command window.
A picture of the bifurcation curves and their bifurcation points computed in *Run 16*, *Run 17* and *Run 18* along with the hyperbola $H = 0$ is given in Figure 12.

An alternative way to compute the fold curves of M^2_{LG}, computed in *Run 17* and *Run 18*, is to switch directly from the GPD point on the `PD` curve, computed in *Run 16*. This is *Run 19*:

```
>> global opt cds fpmds pdmds
>>>>>>> SWITCHING <<<<<<<
opt=contset(opt,'Maxnumpoints',250);
ap = [11 13];y1=x1(1:4,s1(2).index);p1(ap)=x1([5 6],s1(2).index);
[x0,v0]=init_GPD_LP2m(@LeslieGower,1e-3,y1,p1,[11 13],1);
opt=contset(opt,'Backward',0);
[x7,v7,s7,h7,f7]=cont(@limitpointmap,x0,v0,opt);
first point found
tangent vector to first point found
```

```
elapsed time  = 4.4 secs
npoints curve = 250
cpl(x7,v7,s7,[6 5]);
y1=x2(1:4,s2(2).index);p1(ap)=x2([5 6],s2(2).index);
opt=contset(opt,'Maxnumpoints',260);
[x0,v0]=init_GPD_LP2m(@LeslieGower,1e-1,y1,p1,[11 13],1);
[x8,v8,s8,h8,f8]=cont(@limitpointmap,x0,v0,opt);
first point found
tangent vector to first point found
elapsed time  = 4.2 secs
npoints curve = 260
cpl(x8,v8,s8,[6 5]);
```

This test is run by typing *LeslieGower30* in the command window.
Now we return to *Run 1* where a PD point was detected. We compute the flip curve in the PD point, this is *Run 20*:

```
>> global x12 v12 s12 opt cds fpmds pdmds
>>>>>>> flip curve <<<<<<<
p1(fpmds.ActiveParams)=x12(end,s12(4).index);
x1=x12(1:4,s12(4).index);
opt=contset(opt,'Maxnumpoints',15);
[x0,v0]=init_PDm_PDm(@LeslieGower,x1,p1,[11 13],1);
opt=contset(opt,'Backward',0);
[x1,v1,s1,h1,f1]=cont(@perioddoublingmap,x0,v0,opt);
first point found
tangent vector to first point found
label = GPD , x = ( 21.502856 22.996110 0.000000 0.000000 0.275983 0.450625 )
Normal form coefficient of GPD = 2.721284e-006
label = GPD , x = ( 21.502856 22.996110 0.000000 0.000000 0.161087 0.450625 )
Normal form coefficient of GPD = -1.779620e-013
elapsed time  = 0.7 secs
npoints curve = 20
cpl(x2,v2,s2,[6 5]);
```

This test is run by typing *LeslieGower31* in the command window. We note that the vector v2 is not used in the continuation run.
The continuation leads to a vertical straight line of flip points which is depicted in Figure 13, with two GPD bifurcation points. Now we take a point on the straight line of flip points between the two GPD points, where $(c_{yj}, c_{jy}) = (0.450625; 0.2201)$. Then we compute the curve of fixed points of the period 2 map by starting from the PD point. We call this *Run 21*:

```
>> global  x12 v12 s12 opt fpmds cds fpmds
>>>>>> flip curve <<<<<<<<<<<<<
>> x1=x12(1:4,s12(4).index);p1=p;
>> p1(fpmds.ActiveParams)=x12(5,s12(4).index);
>> opt=contset(opt,'MaxNumPoints',1000);
>> opt=contset(opt,'Backward',1);
```

```
>> [x2,v2]=init_PDm_FP2m(@LeslieGower,x1,p1,s12(4),0.01,1);
>> [x32,v32,s32,h32,f32]=cont(@fixedpointmap,x2,[],opt);
first point found
tangent vector to first point found
label = BP, x = ( 21.502856 22.996110 -0.000000 0.000000 0.450625 )
label = LP, x = ( 25.761683 36.839702 -49.164526 1.145490 0.487045 )
normal form coefficient of LP =1.228524e-003
elapsed time  = 5.4 secs
npoints curve = 1000
>>cpl(x32,v32,s32,[1 5])
```

This test is run by typing *LeslieGower32* in the command window.
The continuation leads to a curve of unstable fixed points. We compute the fold curve of M^2_{LG} starting from the LP point of *Run 21*. We refer to this as *Run 22*:

```
>> global x32 v32 s32 opt cds fpmds lpmds
>>>>>>> fold curve <<<<<<<
>> p1=p;p1(fpmds.ActiveParams)=x32(end,s32(3).index);x1=x32(1:4,s32(3).index);
>> opt=contset(opt,'Maxnumpoints',500);
>> [x0,v0]=init_LPm_LPm(@LeslieGower,x1,p1,[11 13],2);
>> opt=contset(opt,'Backward',1);
>> [x33,v33,s33,h33,f33]=cont(@limitpointmap,x0,v0,opt);
first point found
tangent vector to first point found

elapsed time  = 8.1 secs
npoints curve = 500
>> cpl(x33,v33,s33,[6 5]);
>> opt=contset(opt,'Backward',0);
>> opt=contset(opt,'Maxnumpoints',300);
>>[x332,v332,s332,h332,f332]=cont(@limitpointmap,x0,v0,opt);
first point found
tangent vector to first point found

elapsed time  = 5.2 secs
npoints curve = 300
>> cpl(x332,v332,s332,[6 5]);
```

This test is run by typing *LeslieGower33* in the command window.
The curve ends at a GPD point on the flip curve computed in *Run 20*. A picture of this curve and the flip curve of *Run 20* are given in Figure 13.

The fold curve of M^2_{LG} is tangent to the straight line of PD points at a GPD. For a close view see Figure 14.

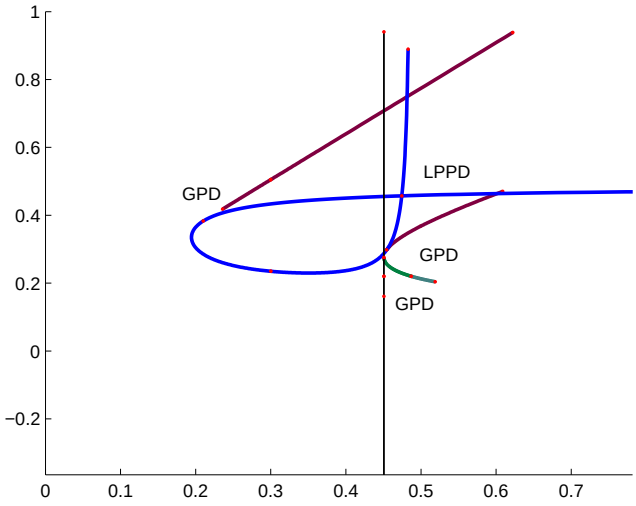


Figure 13: The curves computed in *Run 20* and *Run 22* along with Figure 12 in (c_{yj}, c_{jy}) space.

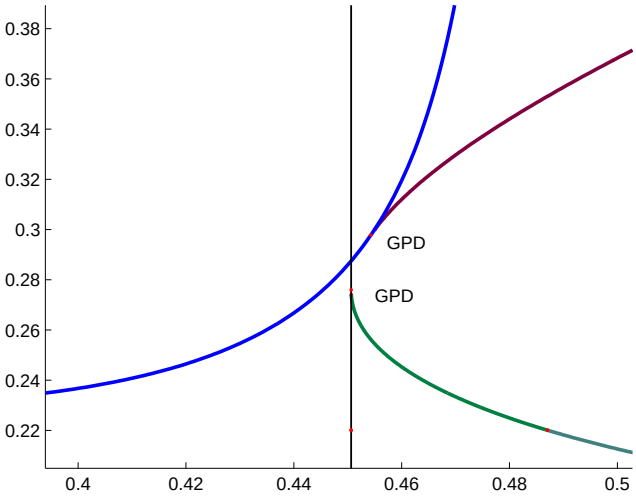


Figure 14: Close up of the tangency of the fold curve of M^2_{LG} to the straight line of `PD` points at a `GPD` point.

Similarly, we can compute a fold curve of M_{LG}^2 that is tangentially born on the horizontal straight line of flip points. We consider *Run 4* where a PD point was detected and then compute the flip curve in this PD point. We call this *Run 23*:

```
>>>>>>> flip curve <<<<<<<
>> global x15 v15 s15 opt cds fpmds pdmds
>> p1(fpmds.ActiveParams)=x15(end,s15(3).index);
>> x1=x15(1:4,s15(3).index);
>> opt=contset(opt,'Maxnumpoints',40);
>> [x0,v0]=init_PDm_PDm(@LeslieGower,x1,p1,[11 13],1);
>> opt=contset(opt,'Backward',0);
>> [x341,v341,s341,h341,f341]=cont(@perioddoublingmap,x0,v0,opt);
first point found
tangent vector to first point found
elapsed time  = 0.6 secs
npoints curve = 40
>> cpl(x341,v341,s341,[6 5]);
>> opt=contset(opt,'Backward',1);
>> opt=contset(opt,'Maxnumpoints',20);
>> [x342,v342,s342,h342,f342]=cont(@perioddoublingmap,x0,v0,opt);
first point found
tangent vector to first point found
label = GPD , x = ( 0.000000 0.000000 32.686981 23.681384 0.170849 0.021904 )
Normal form coefficient of GPD = -9.813682e-009
label = GPD , x = ( 0.000000 0.000000 32.686981 23.681384 0.170849 0.103960 )
Normal form coefficient of GPD = -3.239219e-009

elapsed time  = 0.8 secs
npoints curve = 20
cpl(x342,v342,s342,[6 5]);
```

This test is run by typing *LeslieGower34* in the command window.
The continuation leads to a horizontal straight line of flip points, with two GPD bifurcation points. Now we take a point on the straight line of flip points between the two GPD points, where $(c_{yj}, c_{jy}) = (0.450625; 0.2201)$. Then we compute a curve of fixed points of M_{LG}^2, starting from the PD point. We call this *Run 24*:

```
>>>>> fixed points curve of the second iterate<<<<
>> global x15 v15 s15 opt cds fpmds
>> x1=x15(1:4,s15(3).index);p1=p;
>> p1(fpmds.ActiveParams)=x15(5,s15(3).index);
>> opt=contset(opt,'MaxNumPoints',300);
>> [x2,v2]=init_PDm_FP2m(@LeslieGower,x1,p1,s15(3),0.01,1);
>>[x35,v35,s35,h35,f35]=cont(@fixedpointmap,x2,[],opt);
first point found
tangent vector to first point found
label = LP, x = ( -11.313105 6.463612 54.415142 9.852869 0.167365 )
normal form coefficient of LP =-1.750138e-003
```

```
elapsed time  = 1.7 secs
npoints curve = 300
>> cpl(x35,v35,s35,[1 5])
```

This test is run by typing *LeslieGower35* in the command window. We note that the vector `v2` is not used in the continuation run.

10.3 A Cod Stock model

10.3.1 The model, its fixed points and their stability properties

The roots of the present model can be found in [8, 9, 11]. The model that we use is a two-dimensional difference equation (145) with seven parameters described in [59] as follows

$$\begin{aligned} x_{1,t+1} &= F.e^{-\beta_1 x_{2,t}} x_{2,t} + (1-\mu_1)e^{-\beta_2 x_{2,t}} x_{1,t} \\ x_{2,t+1} &= P.e^{-\beta_3 x_{2,t}} x_{1,t} + (1-\mu_2)x_{2,t} \end{aligned} \tag{145}$$

where $x_{1,t}$ and $x_{2,t}$ are the immature and mature parts of the population at time t respectively. F is the fecundity (that is the number of newborns per adult), and P, $0 < P \leq 1$, is the fraction of the immature population that survive and enter the mature stage one time unit later. μ_2 may be interpreted as natural death rate. μ_1 combines natural death and maturation, so $\mu_1 \geq P$. The corresponding parameters $\beta_i, i = 1, 2, 3$ will be referred to as cannibalism parameters. Thus, F is reduced by the factor $e^{-\beta_1 x_2}$ due to cannibalism practised by the mature part of the population. In a similar way the remaining part of the immature population $(1-\mu_1)$ is reduced by the factor $e^{-\beta_2 x_2}$, and finally the survival from immature stage to the mature stage is reduced by the factor $e^{-\beta_3 x_2}$ due to cannibalism practised by individuals in the mature stage. In the model (145) we do not consider cannibalism within the stages.
The map (145) is implemented in the mapfile *NAFStock.m*.
Clearly, the vector $(x_1^*, x_2^*) = (0, 0)$ is a trivial fixed point of (145). Evaluation of the Jacobian of (145) at the trivial fixed point gives,

$$\begin{pmatrix} 1-\mu_1 & F \\ P & 1-\mu_2 \end{pmatrix} \tag{146}$$

The fixed point is stable if

$$R = \frac{F.P + (1-\mu_1)\mu_2}{\mu_2} < 1 \tag{147}$$

A nontrivial fixed point (x_1^*, x_2^*) of the model must satisfy:

$$\begin{aligned} x_1^* &= F.e^{-\beta_1 x_2^*} x_2^* + (1-\mu_1)e^{-\beta_2 x_2^*} x_1^* \\ x_2^* &= P.e^{-\beta_3 x_2^*} x_1^* + (1-\mu_2)x_2^* \end{aligned} \tag{148}$$

By the second equation of (148), we have

$$x_1^* = \frac{\mu_2}{P} e^{\beta_3 x_2^*} x_2^* \tag{149}$$

where x_2^* must satisfy the nonlinear equation

$$g(x_2^*) = \frac{F.P}{\mu_2} e^{-(\beta_1+\beta_3)x_2^*} + (1-\mu_1)e^{-\beta_2 x_2^*} - 1 = 0 \tag{150}$$

One can show that the nontrivial fixed point is stable when:

$$(1-\mu_1)e^{-\beta_2 x_2^*}(2-(\beta_1-\beta_2+\beta_3)\mu_2 x_2^*)+2(1-\mu_2)+(\beta_1-\beta_3)\mu_2 x_2^*>0 \tag{151}$$

and

$$1+(1-\beta_1 x_2^*)\mu_2-(1-\mu_1)e^{-\beta_2 x_2^*}(1-(\beta_1-\beta_2+\beta_3)\mu_2 x_2^*)>0 \tag{152}$$

hold.

10.3.2 Numerical stability analysis of the model

Following [59] we consider three special parameter ranges of (145). We note that all normal form coefficients in our computations are small in absolute value; this is caused by the exponentials in the definition of the map and does not indicate that the sign of the coefficients cannot be trusted.

10.3.3 Case study 1

We consider the case where the cannibalism pressures on the newborns, immature population and those on the threshold of entering the mature stage are equal, i.e., $\beta_1=\beta_2=\beta_3=\beta$. Thus, the model (145) is rewritten as

$$\begin{aligned} x_{1,t+1} &= F.e^{-\beta x_{2,t}}x_{2,t}+(1-\mu_1)e^{-\beta x_{2,t}}x_{1,t} \\ x_{2,t+1} &= P.e^{-\beta x_{2,t}}x_{1,t}+(1-\mu_2)x_{2,t} \end{aligned} \tag{153}$$

This map is implemented in the mapfile *Rfish.m*.
The nontrivial solution of this model can be expressed by

$$(x_1^*,x_2^*)=(\frac{\mu_2}{\beta.P}K,\frac{1}{\beta}\ln K) \tag{154}$$

where

$$K=\frac{1}{2}(1-\mu_1)+\sqrt{\frac{F.P}{\mu_2}+\frac{(1-\mu_1)^2}{4}}=\frac{1-\mu_1}{2}+\frac{\sqrt{(1-\mu_1)^2+4(R-1)}}{2} \tag{155}$$

We do a numerical bifurcation analysis of (153) by starting from the model parameters $\beta=1, P=0.5, \mu_1=0.5, F=120$ and $\mu_2=0.9$. For the above parameter set the nontrivial fixed point $(x_1^*,x_2^*)=(32.2814,2.1305)$ is computed from (154) and is stable. We do a numerical continuation of fixed points back and forth where F is the free parameter, we refer to this as *Run 1.* We obtain the following MatContM output:

```
>> global cds fpmds;
>> p=[120;0.5;0.5;0.9;1];ap=1;
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt = contset(opt,'Backward',0);
>> opt = contset(opt,'Singularities',1);
```

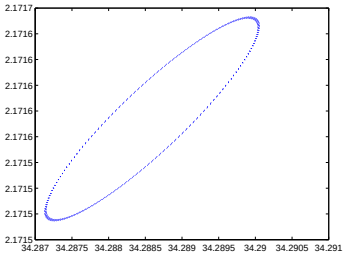


Figure 15: The invariant curve for $\beta_1 = \beta_2 = \beta_3 = 1, \mu_1 = P = 0.5, \mu_2 = 0.9$,$F = 130.62$.

```
>> opt = contset(opt,'MaxNumPoints',300);
>>> curve of fixed points <<<
>> [x0,v0]=init_FPm_FPm(@Rfish,[32.28140;2.1305], p, ap,1);
>> [x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = NS  , x = ( 34.287724 2.171557 130.609334 )
normal form coefficient of NS = -5.721873e-004
elapsed time  = 1.1 secs
npoints curve = 200
>> cpl(x1,v1,s1,[3 1]);
>> opt = contset(opt,'MaxNumPoints',1250);
>>opt = contset(opt,'Backward',1);
>> [x2,v2,s2,h2,f2]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = BP  , x = ( 0.000000 0.000000 0.900000 )
elapsed time  = 3.9 secs
npoints curve = 1250
cpl(x2,v2,s2,[3 1]);
```

This test is run by typing *CodStock1* in the command window.
Two bifurcation points are detected along the fixed point curve, a branch point (BP) and a supercritical Neimark-Sacker point (supercriticality follows from the fact that the normal form coefficient of the `NS` point is negative). The nontrivial fixed point is stable only for $0.9 < F < 130.609334$. The dynamics beyond the upper threshold is a stable invariant curve which surrounds the unstable fixed point. Such a curve is shown in Figure 15.
The new branch of fixed points that was encountered in *Run 1* for $F = 0.9$ is computed in *Run 2* and gives the following MatContM output:

```
>> global x1 v1 s1 h1 f1   fpmds
>> opt = contset;
>>>>> Branch switching at BPm >>>>>>>
>> xx2=x2(1:2,s2(2).index);p1=p;p1(fpmds.ActiveParams)=x2(3,s2(2).index);
>> opt=contset(opt,'Backward',1);
>> opt=contset(opt,'MaxNumPoints',300);
```

```
>> [x2,v2]=init_BPm_FPm(@Rfish,xx2,p1,s2(2),0.0001,1);
>> [x6,v6,s6,h6,f6]=cont(@fixedpointmap,x2,v2,opt);
first point found
tangent vector to first point found
label = BP  , x = ( 0.000000 0.000000 0.900000 )
label = PD  , x = ( 0.000000 0.000000 3.300000 )
normal form coefficient of PD = 8.753732e-002
elapsed time  = 1.3 secs
npoints curve = 300
>> cpl(x6,v6,s6,[3 2])
```

This test is run by typing *CodStock2* in the command window.
Clearly, the new branch is the trivial branch of fixed points. The trivial fixed point is stable before the BP point and unstable afterwards where the basic reproductive number R in (147) becomes larger than 1.
Now we compute the Neimark-Sacker bifurcation curve forth and back, by starting from the NS point of *Run 1*, with two free parameters F and μ_2. We call this *Run 3*:

```
>> global  opt fpmds nsmds cds
>>>>>>>  Fixed points  >>>>
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>>>>>>>  NS curve  starting from the NS point >>>>
>> xx2=x1(1:2,s1(2).index);p1=p;
>> p1(fpmds.ActiveParams)=x1(3,s1(2).index);
>>>>>>>  NS curve  starting from the NS point >>>>
[x2,v2]=init_NSm_NSm(@Rfish,xx2,p1,[1 4],1);
opt=contset(opt,'Singularities',1)
opt=contset(opt,'IgnoreSingularity',[5]);
opt=contset(opt,'MaxNumPoints',2750);
[x31,v31,s31,h31,f31]=cont(@neimarksackermap,x2,v2,opt);
first point found
tangent vector to first point found
label = R3  , x = ( 61.127825 3.001853 399.586101 0.505977 -0.500000 )
Normal form coefficient of R3 : Re(c_1) = -4.134443e-001

elapsed time  = 25.5 secs
npoints curve = 2750
opt = contset(opt,'Backward',1);
opt=contset(opt,'MaxNumPoints',150);
[x32,v32,s32,h32,f32]=cont(@neimarksackermap,x2,v2,opt);
first point found
tangent vector to first point found
label = R2  , x = ( 32.714248 2.069443 117.303643 0.997942 -1.000000 )
Normal form coefficient of R2 : [c , d] = -1.518117e-004, -3.075159e-003
elapsed time  = 2.0 secs
npoints curve = 150
```

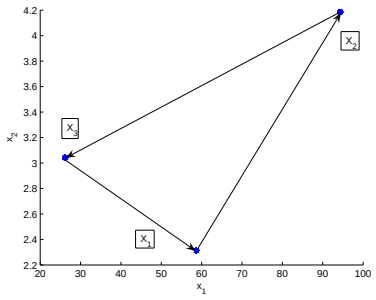


Figure 16: An exact 3-cycle close to the R_3 point, where $F = 399.5861$ and $\mu_2 = 0.444715$.

This test is run by typing *CodStock3* in the command window.
In *Run 3*, we find a resonance 1:3 point. Since its normal form coefficient is negative, the bifurcation picture near the R_3 point is qualitatively the same as presented in [33], Fig. 9.12. In particular, there is a region near the R_3 point where a stable invariant closed curve coexists with an unstable equilibrium. For parameter values close to the R_3 point, the map has a saddle cycle of period three. An exact 3-cycle near the R_3 point is $C_3 = \{X_1, X_2, X_3\}$ where

$$X_1 = (58.66425, 2.31385), X_2 = (94.32305, 4.18521), X_3 = (26.16934, 3.04173)$$

This cycle and the parameter values are given in Figure 16. The multipliers of the fixed point of the third iterate in X_1 are $\lambda_1 = 1.03980469$ and $\lambda_2 = 0.356852$, thus confirming the saddle character. This 3-cycle can be depicted using X_1 and the above values of the parameters and then simulation of the nontrivial fixed point (154) and plotting the resulting points.
If we continue the fixed point of the third iterate of the map starting from X_3 for decreasing values of μ_2 then it gains stability at a fold point for $\mu_2 = 0.444666$. This stable 3-cycle loses its stability again at a `PD` point for $\mu_2 = 0.499060$ after which a series of successive period doubling bifurcations occur such that new orbits of period $3.2^k, k = 1, 2, ...,$ are created. A 6-cycle is given by $C_6 = \{X_1, X_2, X_3, X_4, X_5, X_6\}$ where

$$X_1 = (49.79841, 1.68883), X_2 = (129.26567, 5.43071), X_3 = (9.78778, 2.95507)$$
$$X_4 = (61.74516, 1.70878), X_5 = (1.29237, 6.43133), X_6 = (4.24233, 3.26836)$$

This cycle is depicted in Figure 17. A 12-cycle with parameter values is given in Figure 18.

We note that for $\mu_2 \in [0.444666, 0.499060]$ we have bistability of a fixed point of the map and a fixed point of the third iterate.

We now consider the map near the detected R_2 point computed in *Run 3*. Since the normal form coefficients c and d are both negative, we are precisely in the situation of [33], Fig. 9.10 (case $s = -1$). For a region of parameter values close to the R_2 point the map has an unstable 2-cycle that coexists with a stable closed invariant curve. Crossing a bifurcation curve, the 2-cycle simultaneously undergoes a `NS` bifurcation. By branch switching in the R_2 point, we compute the `NS` branch of the second iterate, which corresponds to $H^{(2)}$ in [33], Fig. 9.10. Further, from the R_2 point a flip curve originates. Computing the flip curve, reveals that a flip bifurcation exists in a small vicinity of the parameter $\mu_2 = 0.997942$. This is consistent

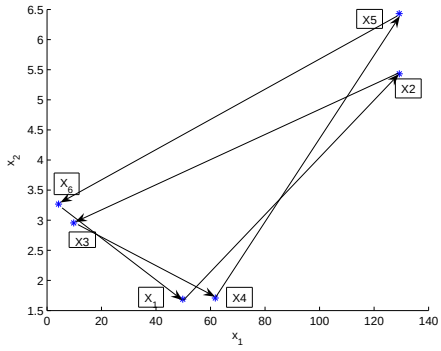


Figure 17: An exact 6-cycle for $F = 399.5861, P = \mu_1 = 0.5, \mu_2 = 0.508$ and $\beta = 1$.

with the analysis in [59] of the reduced model in Case 1. A figure of the Neimark-Sacker curve in *Run 3*, the flip curve through the R_2 point and the branch of `NS` points of the second iterate is given in Figure 19. A magnified picture of these curves is given in Figure 20. This Figure can be compared (qualitatively, of course) with [33], Fig. 9.10.
We now continue the fixed point $(x_1^*, x_2^*) = (15.360; 13.183289)$ along the straight line $F = 114$ with $P = \mu_1 = 0.5, \mu_2 = 0.1$ and $\beta = 1$ and varying μ_2. We note that the fixed point is initially stable, and call this *Run 4*:

```
>> global  opt cds fpmds
>>> curve of fixed points <<<
p=[114;0.5;0.5;0.1;1];ap=4;
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>> opt = contset(opt,'Backward',1);
>> opt = contset(opt,'Singularities',1);
>> opt = contset(opt,'MaxNumPoints',394);
[x0,v0]=init_FPm_FPm(@Rfish,[15.36;13.183289], p, ap,1);
[x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = PD  , x = ( 32.053569 2.055439 0.998335 )
normal form coefficient of PD = 2.952363e-003
elapsed time  = 1.6 secs
npoints curve = 394
```

This test is run by typing *CodStock4* in the command window.
The flip points in Figure 19 below the R_2 point have a positive normal form coefficient. Hence a supercritical stable 2-cycle is born when crossing the flip curve, which coexists with an unstable fixed point of the map.

10.3.4 Case study 2

Now we turn to the case where the cannibalism pressure on the newborn is dominating and decreases as age increases, i.e., $\beta_1 > \beta_2 > \beta_3$. For the numerical stability analysis of the fixed point we consider the parameter set $\mu_1 = \mu_2 = P = 0.5, F = 55$ and $\beta_i = 4 - i, i = 1, 2, 3$. For these parameters the fixed point $(x_1^*, x_2^*) = (2.8213, 1.01868)$ is numerically computed from (149) and (150). We note that it is an unstable fixed point. Now, in *Run 5*, we continue fixed points where F is the free parameter.

```
>> global  opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>>> curve of fixed point <<<
[x0,v0]=init_FPm_FPm(@NAFstock,[2.8213;1.01868], p, ap,1);
[x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = NS  , x = ( 2.718282 1.000000 50.903622 )
normal form coefficient of NS = -3.717346e-002
label = BP  , x = ( -0.000000 -0.000000 0.500000 )
elapsed time  = 2.2 secs
npoints curve = 560
```

This test is run by typing *CodStock5* in the command window.
Run 5 shows that the fixed point is stable for small values of the fecundity, i.e., between BP and NS. When F exceeds the threshold $F_c = 50.903622$, i.e., when the inequality sign in (152) is reversed, we find a stable invariant curve.
Now we continue with free parameter β_1 . We refer to this as *Run 6*:

```
>> global  opt cds fpmds
>> p=[55;0.5;0.5;0.5;3;2;1];ap=5;
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>>> curve of fixed points <<<
[x0,v0]=init_FPm_FPm(@NAFstock,[2.8213;1.01868], p, ap,1);
[x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = NS  , x = ( 3.132638 1.072181 2.793847 )
normal form coefficient of NS = -2.864118e-002
label = PD  , x = ( 60.897776 3.007941 0.332657 )
normal form coefficient of PD = 3.603081e-001
elapsed time  = 5.5 secs
npoints curve = 1560
```

This test is run by typing *CodStock6* in the command window.

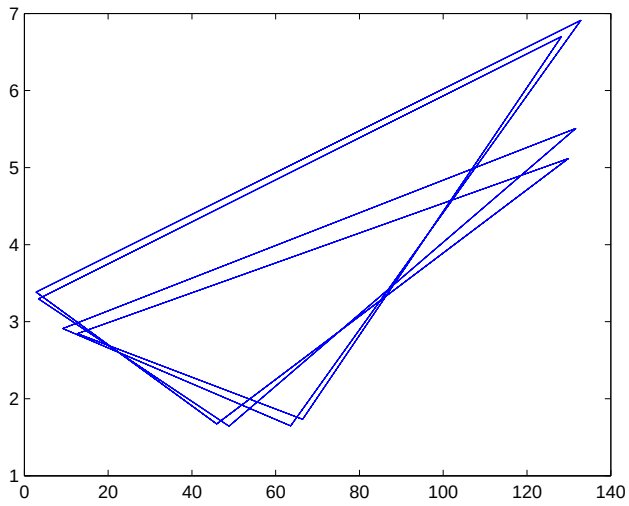


Figure 18: An exact 12-cycle for $F = 399.5861, P = \mu_1 = 0.5, \mu_2 = 0.556571$ and $\beta = 1$.

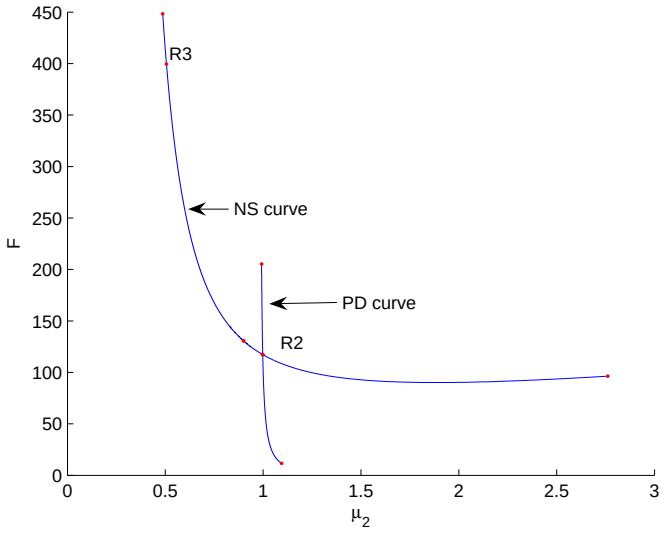


Figure 19: Neimark-Sacker bifurcation curve of *Run 3* and the flip curve through the R_2 point.

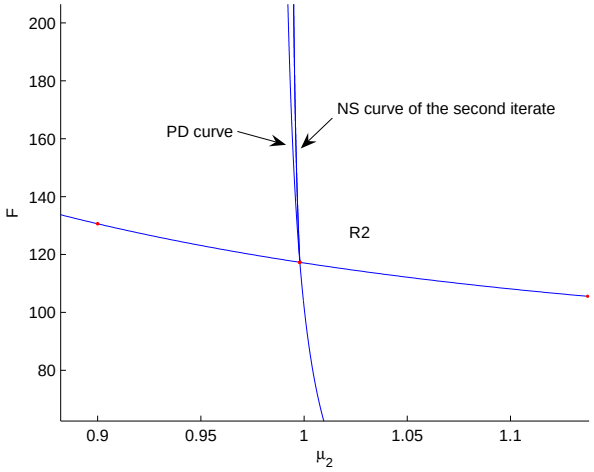


Figure 20: Close up of the flip curve and the NS curve of the second iterate rooted in the R_2 point.

The fixed point is stable between the PD and NS points, i.e., where $\beta_1 \in [2,\ 2.7987], \beta_2 = 2, \beta_3 = 1$. We proceed with the numerical investigation of stability where β_2 is free, in *Run 7*:

```
>> global  opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>>> curve of fixed point <<<
[x0,v0]=init_FPm_FPm(@NAFstock,[2.8213;1.01868], p, ap,1);
[x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = NS   , x = ( 2.932052 1.038207 1.258253 )
normal form coefficient of NS = -3.139363e-002
elapsed time  = 0.8 secs
npoints curve = 50
```

This test is run by typing *CodStock7* in the command window.
We find that the fixed point is unstable before NS and stable afterwards, i.e., where $\beta_1 = 3, \beta_2 \in [1,\ 1.258201], \beta_3 = 1$. Next, we continue with β_3 free, in *Run 8*:

```
>global  opt cds fpmds
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>>> curve of fixed point <<<
>> [x0,v0]=init_FPm_FPm(@NAFstock,[2.8213;1.01868], p, ap,1);
>> [x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,opt);
```

```
first point found
tangent vector to first point found
label = NS  , x = ( 2.926588 1.001660 1.070402 )
normal form coefficient of NS = -3.266591e-002
label = PD  , x = ( 8.864167 0.362921 8.805179 )
normal form coefficient of PD = 1.144210e+000
elapsed time  = 2.0 secs
npoints curve = 500
```

This test is run by typing *CodStock8* in the command window.
By monitoring the multipliers in *Run 8* (stored in `f1`) it is found that the fixed point is stable between the `NS` and `PD` points, i.e., the fixed point is stable where $\beta_1 = 3, \beta_2 = 2, \beta_3 \in [1.070402,\ 2]$. Since the normal form coefficient of the `PD` point is positive, a stable 2-cycle is born where $\beta_3 > 8.805179$. Moreover, it can be seen that increasing β_3, the cannibalism of the immature on the threshold of entering the mature age, results in a wider range of stability than increasing β_1.
For a further analysis we ignore the condition $\beta_1 > \beta_2 > \beta_3$ and compute the Neimark-Sacker curve, by starting at the `NS` point in *Run 6*, with free parameters F and β_3, this is *Run 9*:

```
>>>> NS curve   starting in NSm >>>>>>
>> global x1 v1 s1 opt fpmds nsmds
>>>xx2=x1(1:2,s1(2).index);p1=p;p1(fpmds.ActiveParams)=x1(3,s1(2).index);
>>>[x2,v2]=init_NSm_NSm(@NAFstock,xx2,p1,[1 7],1);
opt=contset(opt,'IgnoreSingularity',[5]);
>>> [x31,v31,s31,h31,f31]=cont(@neimarksackermap,x2,v2,opt);
first point found
tangent vector to first point found
label = R4  , x = ( 3.119219 1.003084 58.799673 1.131015 -0.000000 )
Normal form coefficient of R4 : A = -4.610753e+000 + -1.142472e+000 i
elapsed time  = 1.2 secs
npoints curve = 100
```

This test is run by typing *CodStock9* in the command window.
Since $|A| > 1$ in the R_4 point in *Run 9*, two cycles of period 4 of the map are born. An exact stable 4-cycle for $\beta_3 = 1.131015$ and $F = 58.9$ is given by $C_4 = \{X_1, X_2, X_3, X_4\}$ where
$X_1 = (3.21494, 1.035797); X_2 = (2.93066, 1.01606)$,
$X_3 = (3.031476, 0.97239); X_4 = (3.31453, 0.99085)$
We present this cycle in Figure 21. The multipliers of the fourth iterate of the map in X_1 are $\lambda_1 = 0.999819$ and $\lambda_2 = 0.996348$, confirming the stability of the 4-cycle.

To compute the stability domain of the 4-cycle we note that since $|A| > 1$, there are two half-lines of fold bifurcation curves of the fourth iterate that emanate from the R_4 point. In the next run, *Run 10*, we compute these fold curves and present these curves in Figure 22. The stable fixed points of the fourth iterate exist in the wedge between the two half-lines; they lose their stability when encountering the half-lines. We note that the stable 4-cycles exist in a wide parameter region but there is no bistability with fixed points of the original map.

```
>>> curve of fixed point <<<
```

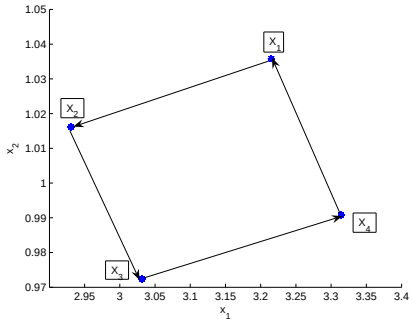


Figure 21: An exact 4-cycle for $\mu_1 = \mu_2 = P = 0.5, \beta_1 = 3, \beta_2 = 2, \beta_3 = 1.131015, F = 58.9$.

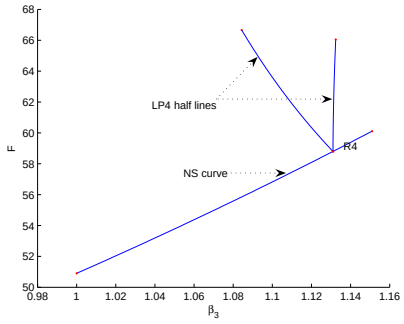


Figure 22: Two half-lines of fold bifurcation points emanate from an R_4 point.

```
first point found
tangent vector to first point found
label = NS   , x = ( 2.718282 1.000000 50.903622 )
normal form coefficient of NS = -3.717346e-002
elapsed time  = 0.6 secs
npoints curve = 50
>>>> NS curve  starting in NSm >>>>>>
first point found
tangent vector to first point found
label = R4   , x = ( 3.119219 1.003084 58.799673 1.131015 -0.000000 )
Normal form coefficient of R4 : A = -4.610753e+000 + -1.142472e+000 i
elapsed time  = 1.9 secs
npoints curve = 200
>>>>>>>>>>>>>>>>switching at R4 to LP41 <<<<<<<<<<
first point found
tangent vector to first point found
elapsed time  = 5.4 secs
npoints curve = 250
>>>>>>>>>>>>>>>>switching at R4 to the second LP41 <<<<<
first point found
tangent vector to first point found
elapsed time  = 5.6 secs
npoints curve = 250
```

This test is run by typing *CodStock10* in the command window.
The Neimark-Sacker curve, starting from the `NS` point in *Run 8*, where μ_2 and β_3 are free parameters is computed in *Run 11*:

```
>>>> NS curve   starting in NSm >>>>>>
>>> xx2=x1(1:2,s1(2).index);p1=p;p1(fpmds.ActiveParams)=x1(3,s1(2).index);
>>> [x2,v2]=init_NSm_NSm(@NAFstock,xx2,p1,[4 7],1);
>>> opt=contset(opt,'IgnoreSingularity',[5]);
>>>[x31,v31,s31,h31,f31]=cont(@neimarksackermap,x2,v2,opt);
```

```
first point found
tangent vector to first point found
label = R4  , x = ( 3.011107 0.988367 0.511112 1.104883 -0.000000 )
Normal form coefficient of R4 : A = -4.675831e+000 + -1.079711e+000 i
label = R3  , x = ( 5.026676 0.735955 0.910954 1.795572 -0.500000 )
Normal form coefficient of R3 : Re(c_1) = -1.581503e+000
label = R2  , x = ( 6.126237 0.627511 1.395201 1.995804 -1.000000 )
Normal form coefficient of R2 : [c , d] = 6.737115e-002, -1.789202e-001
elapsed time  = 1.1 secs
npoints curve = 50
```

This test is run by typing *CodStock11* in the command window.

10.3.5 Case study 3

In the last case we assume $\beta_1 < \beta_2 < \beta_3$. We consider the parameter set $\mu_1 = \mu_2 = P = 0.5, F = 200$ and $\beta_i = i, i = 1, 2, 3$. The fixed point

$$(x_1^*, x_2^*) = (72.8206, 1.33341) \tag{156}$$

is computed from (149) and (150). We note that this is an unstable fixed point. Now we continue the fixed point (156) where F is the free parameter, we call this *Run 12*:

```
> global   opt cds fpmds
>> p=[200;0.5;0.5;0.5;1;2;3];ap=1;
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>>> curve of fixed points <<<
>> [x0,v0]=init_FPm_FPm(@NAFstock,[72.8206;1.33341], p, ap,1);
>> [x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = PD  , x = ( 25.184229 1.056945 64.424674 )
normal form coefficient of PD = 1.818819e-001
elapsed time  = 4.8 secs
npoints curve = 1500
```

This test is run by typing *CodStock12* in the command window.
In *Run 12* there is a supercritical flip bifurcation at the instability threshold, hence a stable 2-cycle is born for $F > 64.424674$. The fixed point is unstable before the `PD` point and stable afterwards. The new branch of fixed points of the second iterate is given in Figure 23.
We proceed with the continuation of fixed points where β_1 is free, we call this *Run 13*:

```
>> global   opt cds fpmds
>> p=[200;0.5;0.5;0.5;1;2;3];ap=5;
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>>> curve of fixed point <<<
>> [x0,v0]=init_FPm_FPm(@NAFstock,[72.8206;1.33341], p, ap,1);
```

```
>>[x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,v0,opt);
first point found
tangent vector to first point found
label = PD  , x = ( 49.555931 1.231597 1.337322 )
normal form coefficient of PD = 4.474137e-003
label = NS  , x = ( 6.567122 0.731554 4.410725 )
normal form coefficient of NS = -7.669904e-003
elapsed time  = 5.0 secs
npoints curve = 1500
```

This test is run by typing *CodStock13* in the command window.
The fixed point is stable between the PD and NS points, i.e., when $\beta_1 \in [1.337322,\ 2], \beta_2 = 2, \beta_3 = 3$. Due to the positive sign of the normal form coefficient of the PD point, a stable 2-cycle coexists with the unstable fixed point of the map for $\beta_1 < 1.337322$.
The fixed point (x_1^*, x_2^*) in (156), remains unstable under variation of the parameter β_2, hence increasing the cannibalism pressure on the immature part is not a stabilizing factor from a dynamical point of view. We now continue with the free parameter β_3, we call this *Run 14*:

```
>> global   opt cds fpmds
>> p=[200;0.5;0.5;0.5;1;2;3];ap=7;
>> opt = contset;
>> opt = contset(opt,'Multipliers',1);
>>> curve of fixed point <<<
>>[x0,v0]=init_FPm_FPm(@NAFstock,[72.8206;1.33341], p, ap,1);
>> [x1,v1,s1,h1,f1]=cont(@fixedpointmap,x0,[],opt);
first point found
tangent vector to first point found
label = PD  , x = ( 64.840130 1.640192 2.241879 )
normal form coefficient of PD = 2.748627e-003
label = NS  , x = ( 29.976635 2.996639 0.768503 )
normal form coefficient of NS = -4.177821e-004
elapsed time  = 3.6 secs
npoints curve = 930
```

This test is run by typing *CodStock14* in the command window.
The fixed point is stable between the PD and NS points, i.e., when $\beta_1 = 1, \beta_2 = 2, \beta_3 \in [2,\ 2.241879]$. From the sign of the normal form coefficient of the PD point, we see that a stable 2-cycle is born when β_3 exceeds the threshold stability $\beta_3 = 2.241879$. An exact stable 2-cycle for $P = \mu_1 = \mu_2 = 0.5, \beta_1 = 1, \beta_2 = 2, \beta_3 = 2.2510715$ and $F = 200$ is given by

$$C_2 = \{X_1, X_2\} = \{(66.459403, 1.69537); (63.349999, 1.578968)\}$$

We proceed with computing the Neimark-Sacker point encountered in *Run 14*, where F and β_3 are free in the continuation, we call this *Run 15*:

```
>>xx2=x1(1:2,s1(3).index);p1=p;p1(fpmds.ActiveParams)=x1(3,s1(3).index);
>>> curve of NS <<<
>>[x2,v2]=init_NSm_NSm(@NAFstock,xx2,p1,[1 7],1);
>> opt = contset;
```

```
>>[x31,v31,s31,h31,f31]=cont(@neimarksackermap,x2,v2,opt);
first point found
tangent vector to first point found
label = R3  , x = ( 36.106260 2.711598 0.582753 0.898279 -0.500000 )
Normal form coefficient of R3 : Re(c_1) = -4.750337e-001
label = R2  , x = ( 49.265517 2.192296 0.835266 1.185576 -1.000000 )
Normal form coefficient of R2 : [c , d] = -5.112120e-004, -1.100939e-003

elapsed time  = 13.5 secs
npoints curve = 400
first point found
tangent vector to first point found
label = R4  , x = ( 16.751436 3.820439 0.354409 0.476980 -0.000000 )
Normal form coefficient of R4 : A = -3.921588e+000 + -2.056128e+000 i

elapsed time  = 8.6 secs
npoints curve = 150
```

This test is run by typing *CodStock15* in the command window.
The R_3 point has the same characteristics (i.e. normal form coefficients with the same sign) as that in *Run 3*. The R_4 point has the same characteristics (absolute value and sign of real and imaginary part) as that in *Run 9*. By the results obtained in *Run 15* there are unstable 3-cycles and stable 4-cycles of fixed points near the R_3 and R_4 points, respectively. We continue by computing the Neimark-Sacker curve forth and back where μ_2 and β_3 are the free parameters, we call this *Run 16*:

```
>>> curve of NS <<<
first point found
tangent vector to first point found
label = R3  , x = ( 36.106260 2.711598 0.582753 0.898279 -0.500000 )
Normal form coefficient of R3 : Re(c_1) = -4.750337e-001
label = R2  , x = ( 49.265517 2.192296 0.835266 1.185576 -1.000000 )
Normal form coefficient of R2 : [c , d] = -5.112120e-004, -1.100939e-003

elapsed time  = 13.5 secs
npoints curve = 400
first point found
tangent vector to first point found
label = R4  , x = ( 16.751436 3.820439 0.354409 0.476980 -0.000000 )
Normal form coefficient of R4 : A = -3.921588e+000 + -2.056128e+000 i

elapsed time  = 8.7 secs
npoints curve = 150
```

This test is run by typing *CodStock16* in the command window.

The R_3 and R_2 points have the same characteristics (i.e. normal form coefficients with the same sign) as those in *Run 3*. The R_4 point has the same characteristics (absolute value and sign of real and imaginary part) as that in *Run 9*.

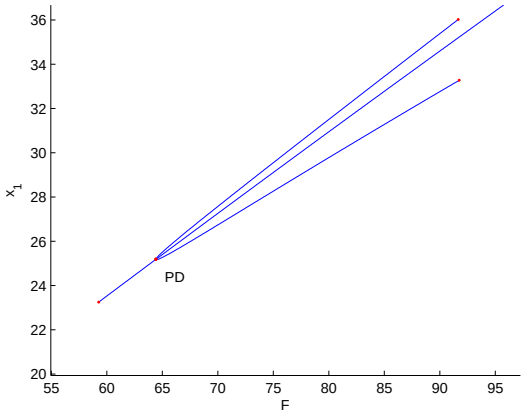


Figure 23: Branch of fixed points of the second iterate and of the original map in (F, x_1) space.

10.4 Heteroclinic and homoclinic connections and tangencies

The examples of heteroclinic and homoclinic connections and tangencies are collected in the directory `Testruns/Connections`.

Examples of the continuation of heteroclinic connections are given in `Hettestrun` and `Hettesthenon`. In `Hettestrun` we consider the generalized Hénon map (GHM)

$$F : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} x_2 \\ \alpha - \beta x_1 - x_2^2 + R x_1 x_2 + S x_2^3 \end{pmatrix}, \tag{157}$$

which appears in numerous theoretical studies of homoclinic bifurcations. The code is the following:

```
global hetds vhet shet
C=[0.46661702380495,0.51950188769804 ,0.48317182408537 ,
0.47311690109635 ,0.6123,0.841195, 1.22990435,0.6419816,0.1347307,
-0.14345697, -0.33379938281268,-0.38062126993587,-0.40424194997464,
-0.41621339121235,-0.42230324914752,-0.42861702380495;
0.46661702380495,0.37639405084666,
0.43915841488454, 0.45696220552142, 0.2067,-0.276064,
-1.3327006,-1.0930203,-0.7998431,-0.6233856,
-0.49516248970159,-0.46255049612568,-0.44591635361118,-
0.43743735402657,-0.43311132511221,-0.42861702380495];
p=[0.3;-1.057;-0.5;0];
ap=[2];
opt = contset;
[x0,v0]=init_Het_Het(@Ghmap,C, p, ap,2);
opt=contset(opt,'MaxNumpoints',30);
opt=contset(opt,'Singularities',1);
```

```
opt=contset(opt,'Backward',1);
opt = contset(opt,'AutDerivative',0);
[xhet,vhet,shet,hhet,fhet]=cont(@heteroclinic,x0,[],opt);
cpl(xhet,vhet,shet,[35 16])
```

We note that the connection consists of 16 mesh points $x_1, \ldots, x_{16}$ in 2D space. The components of the initial points are input via the 2×16 matrix `C` where the first column contains the components of the starting point and the last column contains the components of the endpoint. The continuation vector has 35 components. The first 32 components indicate the coordinates of the mesh points, column by column, the following 2 indicate Y_U and Y_S in the Riccati equations [30], respectively and the last component is the value of the control parameter β. Hence the Figure that is generated by the `cpl` command will represent the second component of the 8th mesh point x_8 as a function of the free parameter β. The output of the run is as follows:

```
first point found
tangent vector to first point found
label = LP  , x = ( 0.450332 0.450332 0.464235 0.427916
0.486124 0.391578 0.542144 0.295162
0.680067 0.035058 0.973257 -0.628904 1.192852
-1.382567 0.417078 -0.981090 -0.036976 -0.709741
-0.254175 -0.571162 -0.355358 -0.504248 -0.402531
-0.472468 -0.424600 -0.457463 -0.434950 -0.450395
-0.439810 -0.447070 -0.444117 -0.444117 -0.004376
-0.005822 -1.009322 )
label = LP  , x = ( 0.471227 0.471227 0.487755
0.443527 0.517117 0.392578 0.597800 0.245433 0.806172
-0.186179 1.203153 -1.234826 0.805668 -1.173190
0.258458 -0.870745 -0.069068 -0.666717 -0.241453
-0.552315 -0.330135 -0.491249 -0.375727 -0.459196 -
0.399232 -0.442484 -0.411379 -0.433796 -0.417667
-0.429288 -0.424423 -0.424423 0.000144 0.000193 -1.070206 )
elapsed time  = 12.9 secs
npoints curve = 30
```

During the continuation two limit points of the connection (heteroclinic tangencies) are found, see Figure 24.

`Hettesthenon` is a test very similar to `Hettestrun`. There are more mesh points (48 instead of 16) and α is chosen as the free parameter instead of β. The graphical output in Figure 25 presents the first component of each mesh point as a function of α.

Examples of the continuation of heteroclinic tangencies are given in `HetTtestrun`, `HetTtestrun2` and `HetTtestrun3D`. In `HetTtestrun` the computations in `Hettestrun` are executed first, then the first discovered tangency is continued forward and backward in two free parameters α and β. A two-parameter picture of the computed branches is presented in Figure 26.

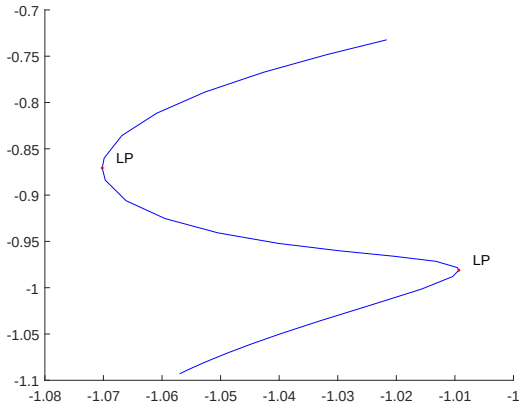


Figure 24: A branch of heteroclinic connections in the Generalized Hénon map. The second component of the 8th mesh point is presented as a function of the free parameter β.

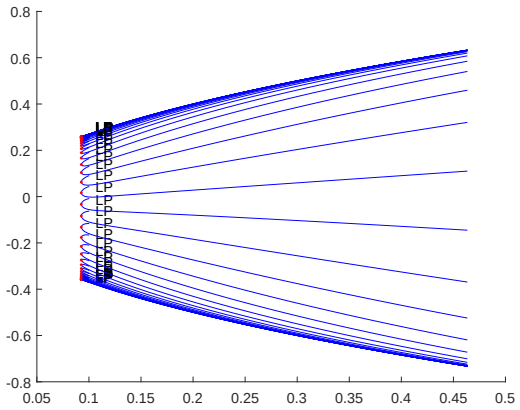


Figure 25: A branch of heteroclinic connections in the Generalized Hénon map. The first component of the each mesh point is presented as a function of the free parameter α.

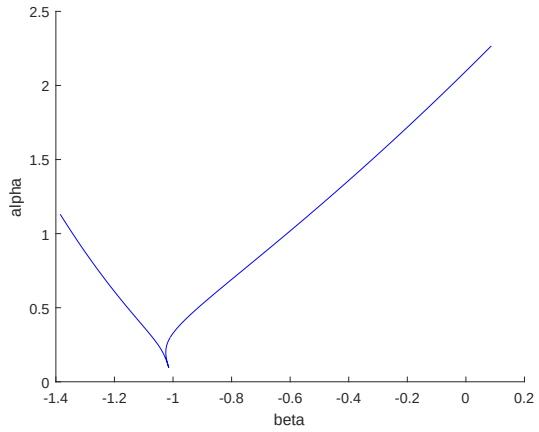


Figure 26: A two-parameter picture of a branch of heteroclinic tangencies in the Generalized Hénon map.

`Hettestrun3D` is a rather artificial 3D test. It uses a 3D extension of (157) with an additional parameter G.

$$F : \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} x_2 \\ \alpha - \beta x_1 - x_2^2 + R x_1 x_2 + S x_2^3 \\ G x_3 \end{pmatrix}, \tag{158}$$

10.5 Computation of one-dimensional invariant manifolds

The test examples on computing invariant manifolds are in the subdirectory `InvManifolds` of the directory `Testruns`.

10.5.1 Heteroclinic connections in 2D space

We consider again the generalized Hénon map (157). In the first run we compute the stable manifold in a saddle fixed point $x1$, the unstable manifold in another saddle fixed point $x0$, and the intersection points of the two manifolds. The run is the following:

```
global opt
opt=contset;
x0=[ .4666170238;  .4666170238];
x1=[-.4286170238; -.4286170238];
p0=[0.3,-1.057,-0.5,0];
optM=init_FPm_1DMan(@Ghmap,x1,p0,2);
% Smanifold
optM.function='Smanifold';
optM.distanceInit=-optM.distanceInit;
optM.eps=1e-10;
optM.nmax=15000;
```

```
    optM.deltaMax=0.001;
    optM.deltak=1e-5;
    optM.deltaAlphaMax=0.001;
    optM.deltaAlphaLowMax=0.0001;

    [a,l]=growman(optM);
    % Umanifold
    optM=init_FPm_1DMan(@Ghmap,x0,p0,2);
    optM.function='Umanifold';
    optM.distanceInit=1e-5;
    optM.eps=1e-10;
    optM.nmax=15000;
    optM.deltaMax=0.001;
    optM.deltak=1e-5;
    optM.deltaAlphaMax=0.001;
    optM.deltaAlphaLowMax=0.0001;

    [b,l]=growman(optM);
    % plot
    figure
    ylim([-2,2]);
    xlim([-2,2]);
    line(x0(1),x0(2),'marker','.','color','k')
    line(x1(1),x1(2),'marker','.','color','k')
    line(a(1,:),a(2,:),'color','b')
    line(b(1,:),b(2,:),'color','r')
    % heteroclinic orbit
%het=findintersections(b,a,2);
het=Projectie2(b,a,2);
    c=het{1};
    d=het{2};
    line(c(1,:),c(2,:),'color','g','marker','.')
    line(d(1,:),d(2,:),'color','m','marker','.')
```

which is executed by typing `Manifolds2DHet` in the command line. We note the following:

- `x0, x1` are two saddle fixed points of (157) for the parameter values in `p0`.
- The call to `init_FPm_1DMan` initializes a computation of invariant manifolds to start from `x1` and sets default values in the structure `optM`. The last entry ("2") sets the number of iterations, not the dimension of the phase space.
- The next 8 lines specify that a stable manifold is to be grown and sets several user-chosen values for the fields of the structure `optM`. `optM` will later be used as the `opt_man` structure in `SManifold`.
- The command

```
[a,l]=growman(optM);
```

computes the manifold. The output `a` is a 2×15000 array whose columns contain the coordinates of 15000 points on the stable manifold of `x1`. The scalar `l` is the arclength of the computed part of the stable manifold.

- Similarly, the command

```
[b,l]=growman(optM);
```

delivers a 2×15000 array `b` whose columns contain the coordinates of 15000 points on the unstable manifold of `x0`. The scalar `l` is the arclength of the computed part of the unstable manifold.

- The output of the command

```
het=Projectie2(b,a,2);
```

is a cell array whose cells contain the orbits discovered among the intersection points of the two computed manifolds `b` and `a`. We note that the unstable manifold is the first input argument, the stable manifold is second. The last argument (2) indicates the iteration number. The output can be inspected in the command window as

```
>> het

het =

  2 x 1 cell array

    [2 x 7 double]
    [2 x 6 double]
```

So two heteroclinic connections of the second iterate of the map are found, consisting of 7 and 6 points, respectively.

- In the next lines, the two connecting orbits are put into the MatContM Figure by green and magenta lines, respectively.

The output is the following:

```
>> Manifolds2DHet
accepted points: 1000;  arc length: 0.63119
accepted points: 2000;  arc length: 1.2712
             ...
accepted points: 15000;  arc length: 9.611
Joint Nmax: 15000.    Manifold length: 9.611
Elapsed time is 11.505337 seconds.
accepted points: 1000;  arc length: 0.6465
accepted points: 2000;  arc length: 1.2864
             ...
```

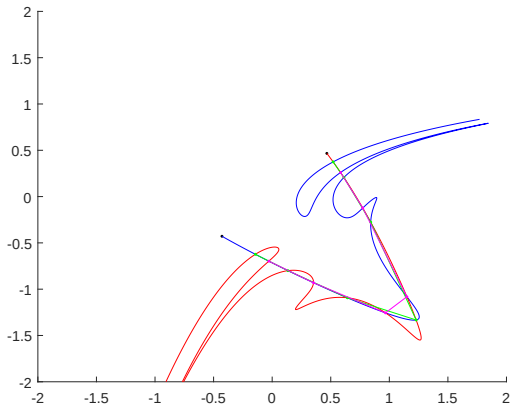


Figure 27: Computation of heteroclinic connections in the Generalized Hénon map.

```
accepted points: 15000;  arc length: 9.6701
Joint Nmax: 15000.    Manifold length: 9.6701
Elapsed time is 3.816045 seconds.
Elapsed time is 2.105612 seconds.
Elapsed time is 2.108011 seconds.
>>
```

After each manifold computation the number of computed points and the arclength of the manifold are given, as well as the elapsed time. At the end of the output the elapsed time span in the intersection algorithm `Projectie2.m`, i.e. the time spent to find all intersection points plus the time spent to compute the connections, is given. The graphical output is seen in Figure 27.

10.5.2 A homoclinic connection in 2D space and its continuation

We consider again the generalized Hénon map (157). In the next run we compute the stable and unstable manifolds in a saddle fixed point `x0,` the intersection points of the two manifolds and the orbits that connect these points. We then continue an (approximation to a) homoclinic orbit under variation of a parameter and discover two homoclinic tangencies. The run is the following:

```
clear all
global opt
opt=contset;
x0=[-1.621146385;-1.621146385];
p0=[-0.4,1.03,-0.1,0];
optM=init_FPm_1DMan(@Ghmap,x0,p0,1);
% Umanifold
optM.function='Umanifold';
optM.distanceInit = 1e-5;
```

```
    optM.distanceInit=-optM.distanceInit;
    optM.eps=1e-10;
    optM.nmax=20000;
    optM.deltaMax=0.011;
    optM.deltak=1e-5;
    optM.deltaAlphaMax=0.001;
    optM.deltaAlphaLowMax=0.0001;

    [a,l]=growman(optM);
    % Smanifold
    optM.function='Smanifold';
    % n.b. some parameters are used only here, ex: NtwMax
    [b,l]=growman(optM);
    % plot
    figure
    ylim([-2,1]);
    xlim([-2,1]);
    line(x0(1),x0(2),'marker','.','color','k')
    line(a(1,:),a(2,:),'color','r')
    line(b(1,:),b(2,:),'color','k')
    % homocline curves
    %hom=findintersections(a,b,1);
hom=Projectie2(a,b,1);
    % plot some homoclinic curves
    % the primaries (the 2 longest curves)
    c=hom{4};
    d=hom{end};
    line(c(1,:),c(2,:),'color','g','marker','.')
    line(d(1,:),d(2,:),'color','m','marker','.')
    % some secondaries
    %e=hom{12};
    %f=hom{13};
    %g=hom{14};
    %h=hom{15};
    %line(e(1,:),e(2,:),'color','g','marker','.')
    %line(f(1,:),f(2,:),'color','m','marker','.')
    %line(g(1,:),g(2,:),'color','c','marker','.')
    %line(h(1,:),h(2,:),'color','y','marker','.')
    %% continuation of the primary curve
    opt=contset;
    opt=contset(opt,'MaxNumPoints',100);
    opt=contset(opt,'Singularities',1);
    opt=contset(opt,'MinStepsize',1e-8);
    [x0hom,v0hom]=init_Hom_Hom(@Ghmap,[x0,c],p0,2,1);
    [xhomc1,vhomc1,shomc1,hhomc1,fhomc1]=cont(@homoclinic,x0hom,[],opt);
    opt=contset(opt,'Backward',1);
    [xhomc2,vhomc2,shomc2,hhomc2,fhomc2]=cont(@homoclinic,x0hom,[],opt);
```

```
figure
    cpl(xhomc1,vhomc1,shomc1,[size(xhomc1,1),14])
    cpl(xhomc2,vhomc2,shomc2,[size(xhomc2,1),14])
    figure
    cpl(xhomc1,vhomc1,shomc1,[size(xhomc1,1),1])
    cpl(xhomc2,vhomc2,shomc2,[size(xhomc2,1),1])
```

and the code is executed by typing `Manifolds2DHom` in the command line. Comparing this code to the one in §10.5.1 we note the following:

- The iteration number is 1 (hence the last argument in the call to `intersections.m` is 1.
- `hom` is a cell array and each cell is a matrix whose columns constitute an orbit of the map. The largest ones are `c=hom{4}` with 12 columns and `d=hom{end}` with 13 columns.
- `c` is used to start a continuation of homoclinic connections under variation of the second system parameter, i.e. β.
- The continuation vector has 29 components, namely $26 = 2 \times 13$ for the coordinates of the points of the connection (starting with the saddle fixed point), 2 for the auxiliary variables in the Riccati equations and finally the parameter β.

The output of the run is as follows:

```
>> Manifolds2DHom
accepted points: 1000;  arc length: 9.5926
accepted points: 2000;  arc length: 19.7458
                 ...
accepted points: 20000;  arc length: 207.2231
Joint Nmax: 20000.    Manifold length: 207.2231
Elapsed time is 4.821864 seconds.
accepted points: 1000;  arc length: 9.6978
accepted points: 2000;  arc length: 19.9522
                 ...
accepted points: 20000;  arc length: 205.7572
Joint Nmax: 20000.    Manifold length: 205.7572
Elapsed time is 12.191296 seconds.
Elapsed time is 4.131338 seconds.
Elapsed time is 4.143104 seconds.
first point found
tangent vector to first point found
label = LP  , x = ( -1.586184 -1.586184 -1.577441 -1.559458 -1.559458 -1.505227
-1.505227 -1.345694 -1.345694 -0.912769 -0.912769 -0.014347 -0.014347 0.508497
0.508497 -0.643536 -0.643536 -1.288377 -1.288377 -1.501234 -1.501234 -1.562632
-1.562632 -1.579706 -1.579706 -1.584408 0.001557 0.000713 0.996980 )
label = LP  , x = ( -1.704645 -1.704645 -1.693470     ...
label = LP  , x = ( -1.586178 -1.586178 -1.559718     ...
label = LP  , x = ( -1.704628 -1.704628 -1.668280     ...
```

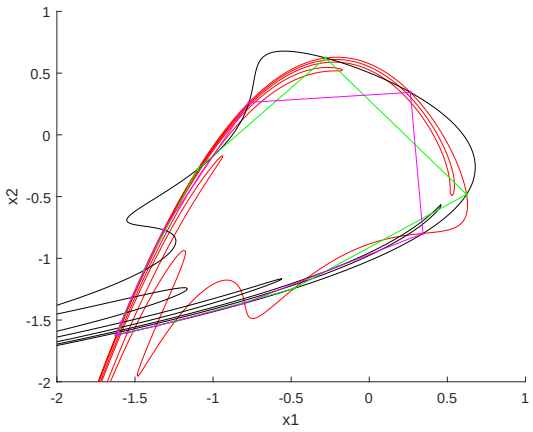


Figure 28: Computation of homoclinic connections in the Generalized Hénon map.

```
label = LP   , x = ( -1.585974 -1.585974 -1.507525     ...
label = LP   , x = ( -1.704037 -1.704037 -1.589365     ...    )

elapsed time  = 1.3 secs
npoints curve = 100
first point found
tangent vector to first point found
label = LP   , x = ( -1.704646 -1.704646 -1.701241 -1.693428 -1.693428 -1.667816
-1.667816 -1.584737 -1.584737 -1.324817 -1.324817 -0.606406 -0.606406 0.622168
0.622168 -0.076398 -0.076398 -1.091542 -1.091542 -1.515021 -1.515021 -1.649305
-1.649305 -1.688765 -1.688765 -1.700118 -0.000016 -0.000008 1.109763 )
label = LP   , x = ( -1.586185 -1.586185 -1.583315    ...
label = LP   , x = ( -1.704649 -1.704649 -1.703615    ...
label = LP   , x = ( -1.586195 -1.586195 -1.585255    ...
label = LP   , x = ( -1.704783 -1.704783 -1.704469    ...
label = LP   , x = ( -1.586555 -1.586555 -1.586248    ...     )

elapsed time  = 0.8 secs
npoints curve = 100
```

The code also generates three figures. In Figure 28 the two manifolds and two homoclinic connections (`c` and `d`) are shown. In Figure 29 the continuation of the homoclinic connection is illustrated by plotting the second coordinate of the seventh point versus the free parameter β. Apparently a large number of limit points (i.e. homoclinic tangencies) is discovered. However, by plotting in Figure 30 the first coordinate of the saddle fixed point versus the free parameter we see that there are really only two tangencies and the homoclinic connection moves forward and backward between the two. The illusion in Figure 29 is created by the fact that as a side-effect of the continuation the orbit points shift along the manifolds. (the choice of the second coordinate of the seventh point is arbitrary)

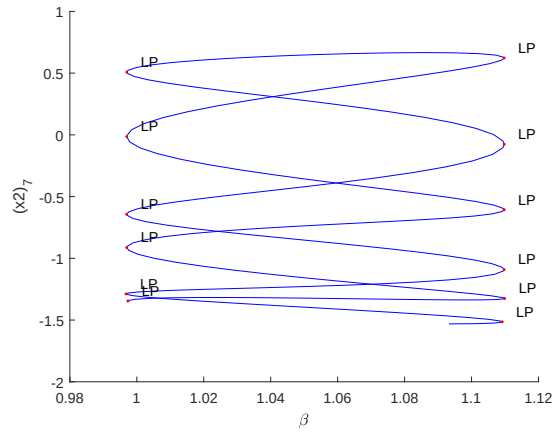


Figure 29: Continuation of a homoclinic connection: second coordinate of seventh point versus parameter β.

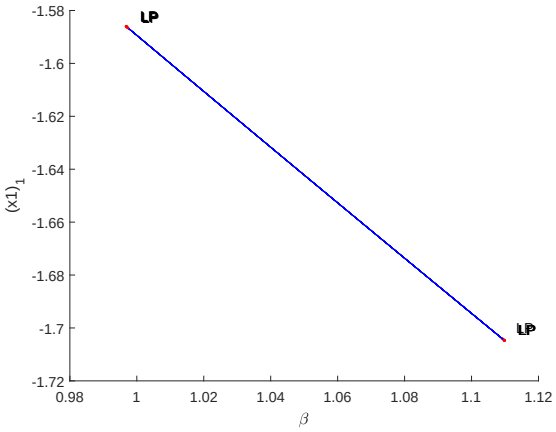


Figure 30: Continuation of a homoclinic connection: first coordinate of saddle point versus parameter β.

10.5.3 The unstable manifold of the 3D Euler-Lorenz map

The Euler-Lorenz map is a 3D map with parameters σ, r, b, h given by

$$F : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x + h\sigma(y - x) \\ y + h(rx - y - xz) \\ z + h(xy - bz)) \end{pmatrix}. \tag{159}$$

Formally, this map arises when numerically solving the Lorenz equations (with parameters σ, r, b) by using the forward Euler method with steplength h. The origin is always a fixed

point of the map and we will use standard parameter values for which the origin has a one-dimensional unstable manifold and a two-dimensional stable manifold. We consider the run:

```
%% Euler-Lorenz test Umanifold
    clear all
    %
    global opt
    opt=contset;

    x0=[0,0,0]';
    sig=10;r=8.37;b=8/3;h=0.1;
    p0=[sig,r,b,h];

    optM=init_FPm_1DMan(@EulerLorenz,x0,p0,1);
    optM.NwtMax=50;
    optM.nmax=60000;
    optM.deltaMax=0.11;
    optM.Arc=500;
    optM.distanceInit=1e-4;
    optM.searchListLength=100;
    [M,l]=growman(optM);
    % plot
    figure
    plot3(M(1,:),M(2,:),M(3,:))
```

This run is executed by typing `Manifold3DU` in the command line. We note that it is not needed to specify that the unstable manifold is to be computed.

The output of the run is

```
>> Manifold3DU
Default chosen manifold: UManifold, with initial direction: -0.56258    -0.82674
To change those values redefine "function" and "direction"
accepted points: 1000;  arc length: 55.854
accepted points: 2000;  arc length: 98.9021
accepted points: 3000;  arc length: 122.8654
WARNING: Recoverable ConvErr:NoConvergence
Warning: Recoverable ConvErr:NoConvergence
> In warnconsole (line 3)
  In UManifold>findSegment (line 158)
  In UManifold (line 38)
  In growman (line 36)
  In Manifold3DU (line 19)
accepted points: 4000;  arc length: 159.0229
                         ...
accepted points: 10000;  arc length: 467.579
Joint maximum searched manifold length: 500.0451    in: 10626 points.
Elapsed time is 2.382000 seconds.
```

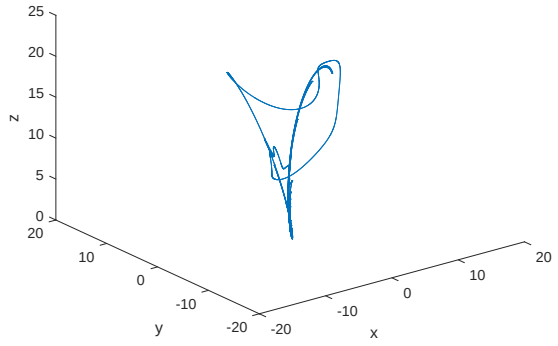


Figure 31: The unstable manifold of the Euler-Lorenz map.

The warning about a recoverable convergence error does not mean that the algorithm failed. Still, a large number of such warning is an indication that there might be problems; preferably the computations should be redone with tighter convergence bounds. The picture of the unstable manifold is given in Figure 31.

References

[1] Allgower, E.L., and Georg, K., Numerical Continuation Methods: An introduction, Springer-Verlag, (1990).

[2] D. R. Aronson, M. Chory, G. Hall, and R. McGehee, Bifurcations from an invariant circle for two-parameter families of maps on the plane: A computer assisted study, Comm. Math. Phys., 83 (1982), 303–354.

[3] Beyn, W.J., Champneys, A., Doedel, E., Govaerts, W., Kuznetsov, Yu.A., and Sandstede, B., Numerical continuation and computation of normal forms. In: Fiedler, B., Iooss, G., and Kopell, N., (eds.) "Handbook of Dynamical Systems : Vol 2", Elsevier 149–219, (2002).

[4] W. -J. Beyn, T. Hüls, J.-M. Kleinkauf, and Y.-K. Zou, Numerical analysis of degenerate connecting orbits of maps, Int. J. of Bifurcation and Chaos, 14 (10), (2004) 3385-3407.

[5] W.-J. Beyn and J.-M. Kleinkauf, The numerical computation of homoclinic orbits for maps. SIAM J. Numer. Anal., 34 (1997), 1207-1236.

[6] Brauer, F., and Castillo-Chávez, C., Mathematical Models in Population Biology and Epidemiology, Springer-Verlag (2000).

[7] H. Broer, R. Roussarie, and C. Simó, Invariant circles in the Bogdanov-Takens bifurcation for diffeomorphisms, Ergodic Theory Dynamical Systems, 16 (1996) 1147-1172.

[8] Caswell, H., Matrix Population Models, Sunderland, Massachusetts: Sinaur Ass. Inc. Publishers, (2001).

[9] Cushing, J.M., Costantino, R.F., Dennis, B., Deshamias, R.A., and Henson, A.M., Nonlinear population dynamics: models, experiments and data. J. Theor.Biol. 194, 1-9 (1998).

[10] J. W. Demmel, L. Dieci, and M. J. Friedman, Computing connecting orbits via an improved algorithm for continuing invariant subspaces, Siam J. Sci. Comput., Vol. 22, No. 1, (2000) 81-94.

[11] Dennis, B., Deshamais, R.A., Cushing, J. M., and Costantino, R.F., Transition in population dynamics: equilibria to periodic cycles to aperiodic cycles . J. Anim. Ecol.6b, 704-729, (1997).

[12] Dhooge, A., Govaerts., W., and Kuznetsov, Yu. A., MatCont: A Matlab package for numerical bifurcation analysis of ODEs, ACM Transactions on Mathematical Software 29(2), pp. 141-164 (2003).

[13] Dhooge, A.,Govaerts, W.,Kuznetsov, Yu.A., Mestrom W. and Riet A.: CL_Matcont: A continuation toolbox in Matlab, (2004) `http://allserv.UGent.be/~adhooge/`

[14] L. Dieci, and M. J. Friedman, Continuation of invariant subspaces, Numer. Linear Algebra Appl., 8 (2001) 317-327.

[15] Doedel, E.J., Champneys, A.R., Fairgrieve, T.F., Kuznetsov, Yu.A., Sandstede, B., and Wang, X.J., AUTO97-AUTO2000 : Continuation and Bifurcation Software for Ordinary Differential Equations (with HomCont), User's Guide, Concordia University, Montreal, Canada (1997-2000). (`http://indy.cs.concordia.ca`).

[16] Doedel, E.J., Govaerts W., and Kuznetsov, Yu.A., Computation of Periodic Solution Bifurcations in ODEs using Bordered Systems, to appear in SIAM Journal on Numerical Analysis, (2001).

[17] Edmunds, J., Cushing, J.M., Costantino, R.F., Henson, S.M., Dennis, B., and Desharnais, R.A., Park's Tribolium competition experiments: a non-equilibrium species coexistence hypothesis, J. Animal Ecology,72,pp.703-712, (2003).

[18] J. P. England, B. Krauskopf and H. M. Osinga, Computing one-dimensional stable manifolds and stable sets of planar maps without the inverse, Siam J. Applied Dynamical Systems, Vol. 3, No. 2, 2004, pp. 161-190.

[19] Fox, J.D., Cannibalism in natural populations. Ann. Rev. Ecol.Syst. 6, 87-106, (1975).

[20] N. K. Gavrilov and L. P. Shilnikov, On three-dimensional systems close to systems with a structurally unstable homoclinic curve: I, Math. USSR-Sb., 17 (1972) 467-485.

[21] N. K. Gavrilov and L. P. Shilnikov, On three-dimensional systems close to systems with a structurally unstable homoclinic curve: II, Math. USSR-Sb., 19 (1973) 139-156.

[22] Govaerts, W., Numerical Methods for Bifurcations of Dynamical Equilibria, SIAM, Philadelphia, (2000).

[23] Govaerts, W., and Khoshsiar Ghaziani, R., Numerical Continuation of fixed points of maps in Cl_MatCont, Proceedings of the Fifth Euromech Nonlinear Dynamics Conference, Eindhoven. Eds. D.H. van Campen, M.D. Lazurko and W.P.J.M. van den Oever, pp. 1740-1749. ISBN 90 386 2667 3, (2005).

[24] Govaerts, W., and Khoshsiar Ghaziani, R., Numerical bifurcation analysis of a nonlinear stage structured cannibalism population model, to appear in J. Diff. Eqns. Appl.

[25] Govaerts, W., Khoshsiar Ghaziani, R., Kuznetsov, Yu. A., and Meijer, H.G.E., Bifurcation analysis of periodic orbits of maps in Matlab, preprint, (2006).

[26] Griewank, A., 2000, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Frontiers in Applied Mathematics, No. 19. SIAM, Philadelphia, PA.

[27] V. G. Gruzdev, and Ju. I. Neimark, A symbolic description of motion in the neighborhood of a not structurally stable homoclinic structure and of its change in transition to close systems, In: System Dynamics, Vol. 8, Gorkii State University, Gorkii, 1975, pp. 13–33. In Russian.

[28] P. Holmes, and J. Guckenheimer, Nonlinear Dynamical Systems and Bifurcations of Vector Fields, Springer-Verlag, 1983.

[29] T. Hüls, Bifurcation of connecting orbits with one nonhyperbolic fixed point for maps, SIADS, Vol. 4, No. 4, 2005, 985-1007.

[30] R. Khoshsiar Ghaziani, W. Govaerts, Yu. A. Kuznetsov, and H.G. E Meijer, Numerical continuation of connecting orbits of maps in MATLAB, J. Diff. Eqns. Appl. 15:8 (2009) 849-875. DOI: 10.1080/10236190802357677.

[31] B. Krauskopf, and H. M. Osinga, Growing 1D and quasi 2D unstable manifolds of maps, J. Comp. Physics 146 (1998) 404-419.

[32] B. Krauskopf, and H. M. Osinga, Growing unstable manifolds of planar maps, preprint, 1997.

[33] Kuznetsov, Yu.A., Elements of Applied Bifurcation Theory, 3rd edition, Springer-Verlag, New York, (2004).

[34] Yu. A. Kuznetsov and V.V. Levitin, CONTENT: Integrated Environment for analysis of dynamical systems. CWI, Amsterdam (1997): `ftp://ftp.cwi.nl/pub/CONTENT`

[35] Yu. A. Kuznetsov and H.G.E. Meijer, Numerical normal forms for codim 2 bifurcations of maps with at most two critical eigenvalues, SISC 26(6), 1932-1954, (2005).

[36] Yu. A. Kuznetsov and H.G.E. Meijer, Remarks on interacting Neimark-Sacker bifurcations, J. Diff. Eqns and Appl. 12(10) 2006, pp. 1009-1035.

[37] Leslie, P.H., and Gower, J.C., The properties of a stochastic model for two competing species, Biometrika, 45, pp.316-330, (1958).

[38] Leslie, P.H., Park, T., and Mertz, D.B., The effect of varying the initial numbers on the outcome of competition between two Tribolium species, J. of Animal Ecology 37,pp.9-23, (1968).

[39] Linehan, J. E., Gregpy, R.S., and Schneider, D.C., Predation risk of age-0 cod(Gadus) relative to depth and substrate in coastal water. J. Exp. Mar. Biol. Ecol. 263. 25-44, (2001).

[40] MATLAB, The Mathworks Inc., `http://www.mathworks.com`

[41] Mestrom, W., Continuation of limit cycles in MATLAB, Master Thesis, Mathematical Institute, Utrecht University, The Netherlands, (2002).

[42] Murray, J.D., Mathematical Biology, 2nd corrected edition, Berlin, Heidelberg, New York: Springer, (1993).

[43] Myers, R.A., Blanchard, W., and Thompson, K. R., Summary of North Atlantic fish recruitment 1942-1987. Can.Tech.Rep.Fish, (1990). & Aquat.Sci.1743.

[44] Ju. I. Neimark, Motions close to doubly-asymptotic motion, Soviet Math. Dokl., 8 (1967) 228-231.

[45] 124. N. Neirynck, B. Al-Hdaibat, W. Govaerts, Yu. A. Kuznetsov and H.G.E. Meijer, Using MatContM in the study of a nonlinear map in economics.
Edited by: Gelfreich, V; FournierPrunaret, D; LopezRuiz, R; et al.
Conference: 5th International Workshop on Nonlinear Maps and their Applications (NOMA) Location: Univ Coll Dublin, Sch Elect and Elect Engn, Dublin, IRELAND Date: JUN 15-16, 2015.
NOMA15 INTERNATIONAL WORKSHOP ON NONLINEAR MAPS AND APPLICATIONS
Book Series: Journal of Physics Conference Series Volume: 692 Article Number: 012013 Published: 2016

[46] N. Neirynck, W. Govaerts, Yu.A. Kuznetsov and H.G.E. Meijer, Numerical bifurcation analysis of homoclinic orbits embedded in one-dimensional manifolds of maps, to appear in ACM Transactions on Mathematical Software.

[47] Ju. I. Neimark, On some cases of periodic motions depending on parameters, Dokl. Akad. Nauk SSSR., 129 (1959) 736-739. In Russian.

[48] S. Newhouse, J. Palis, and F. Takens, Bifurcations and stability of families of diffeomorphisms, Inst. Hautes Etudes Sci. Publ. Math., 57 (1983) 5-71.

[49] Ottersen, G., Environmental impact on variability in recruitment, larval growth and distribution of Arcto-Norwegian cod, Dr Scient thesis, Geophysical Institute, University of Bergen, (1996).

[50] J. Palis, and W. De Melo, Geometric Theory of Dynamical Systems, Springer-Verlag, 1982.

[51] J. Palis, and F. Takens, Hyperbolicity and Sensitive Chaotic Dynamics at Homoclinic Bifurcations: Fractal Dimensions and Infinitely Many Attractors, Cambridge University Press, Cambridge, 1993.

[52] Polis, G. A., The evolution and dynamics of intraspecific predation, Ann. Rev. Ecol.Sys. 12, 225-251, (1981).

[53] Riet, A., A Continuation Toolbox in MATLAB, Master Thesis, Mathematical Institute, Utrecht University, The Netherlands, (2000).

[54] R. Sacker, On invariant surfaces and bifurcation of periodic solutions of ordinary differential equations, Report IMM-NYU 333, New York University, 1964.

[55] R. Sacker, A new approach to the perturbation theory of invariant surfaces, Comm. Pure Appl. Math., 18 (1965), 717–732.

[56] P. Shil'nikov, On a Poincaré–Birkhoff problem, Math. USSR-Sb. 3 (1967) 353-371.

[57] S. Smale, Diffeomorphisms with many periodic points, In: S. Carins, ed., Differential and Combinatorial Topology, Princeton University Press, Princeton, NJ, 1963, pp. 63–80.

[58] Song, Yuhui Lisa., The Juvenile-Adult Leslie/Gower Competition Model, preprint, (2004).

[59] Wikan, A., Eide, A., An analysis of a nonlinear stage-structured cannibalism model with application to the Northeast Arctic cod stock, Bulletin of Mathematical Biology 66, 1685-1704, (2004).

[60] Wikan, A., and Mjølhus, E., Overcompensatory recruitment and generation delay in discrete age-structured population models. J. Math. Biol. 35, 195-239, (1996).

[61] S. Wiggins, Global Bifurcations and Chaos, Springer-Verlag, 1988.

[62] K. Yagasaki, Numerical detection and continuation of homoclinic points and their bifurcations for maps and periodically forced systems, Internat. J. Bifur. Chaos Appl. Sci. Engrg 8 (1998) 1617-1627.

[63] Y. Zhiping, E. J. Kostelich, and J. A. Yorke, Calculating stable and unstable manifolds, Int. J. Bifur. Chaos, 1 (1991) 605–623.

[64] Y. Zhiping, E. J. Kostelich, and J. A. Yorke, Erratum: "Calculating stable and unstable manifolds", Int. J. Bifur. Chaos , 2 (1992), 215.