Description of the extension ddebiftool_nmfm

Maikel Bosschaert, Bram Wage and Yuri Kuznetsov

16 September 2015

This addendum describes the normal form extension to DDE-BifTool, a bifurcation analysis toolbox running in `Matlab`[1] or `GNU Octave`[2]. For systems of DDEs with constant delays this extension is able to automatically detect and accurately locate fold and Hopf bifurcations on steady-state branches, as well as generalized Hopf, double-Hopf, fold-Hopf, Bogdanov-Takens and cusp bifurcations on Hopf and/or fold branches. In addition, the critical normal from coefficients of the bifurcations are computed. This will be illustrated with three different example systems.

Contents

[1] http://www.mathworks.com
[2] http://www.gnu.org/software/octave

1 Overview

Users unfamiliar with DDE-BifTool should look at the demo in the folder `minimal_demo` (script `rundemo.m`). In Section 7 a model of two interacting layers of neurons will be used to demonstrate the detection of and normal form coefficient computation for Hopf, generalized Hopf, double-Hopf and fold-Hopf bifurcations. In Section 8 a system exhibiting a Bogdanov-Takens bifurcation is considered. Here, the sign of the critical normal form coefficients is verified by looking at the stability of the cycles. Lastly, in Section (9) a system with cusp and Bogdanov-Takens is presented.

In version 3.1 of DDE-BifTool a normal form extension, written by Bram Wage [Wage, 2014], was supplied. Here we give a short list of changes and additions that have been made:

1. The normal form extension is now able to detect, locate and compute critical normal form coefficients in the following situations

- Hopf encountered along steady-state curves
- Generalized Hopf, fold-Hopf and double-Hopf encountered along Hopf curves
- Fold encountered along steady-state curves
- Cusp, Bogdanov-Takens and fold-Hopf encountered along Fold curves
- Bogdanov-Takens encountered along Hopf curves

Previously only the first two items were supported. In other words, all codimension-1 bifurcations typically encountered on steady-state curves and codimension-2 bifurcations typically encountered on either fold or Hopf curves are now implemented.

2. The function to detect bifurcations, `br_bifdet.m`, has been completely rewritten and is now supporting

- Detection of multiple bifurcations occurring in the same interval on a bifurcation curve
- Plotting of the test function used to detect the bifurcation
- Realtime monitoring of the eigenvalues

3. The code to locate codimension-2 bifurcation has been separated into a new function `locate_codim2`, instead of being repeated for every bifurcation detected. In the previous version, the middle point selected in the bisection method was not correct in the generalized Hopf and fold-Hopf cases; this has been fixed.

4. The test functions indicating Hopf encountered along steady-state curves and fold-Hopf encountered along Hopf curves have been changed. Now the test function for Hopf uses the same function as for the double-Hopf bifurcation. See (14) for the new test function used to detect fold-Hopf.

5. Abbreviations used for codimension-2 bifurcation have been changed to mimic those being used in MatCont[3].

2 Delay differential equations

Consider a system of delay differential equations with constant delays (DDE),

$$\dot{x}(t) = f(x(t-\tau_0), x(t-\tau_1), \ldots, x(t-\tau_m), \alpha), \tag{1}$$

[3] https://sourceforge.net/projects/matcont/

where $x(t) \in \mathbb{R}^n$, $f : \mathbb{R}^{n(m+1)} \times \mathbb{R}^p \to \mathbb{R}^n$ is a smooth function depending on a number of parameters $\alpha \in \mathbb{R}^p$, and delays τ_i, $i = 0, \ldots, m$, ordered such that $0 = \tau_0 \leq \tau_1 \leq \cdots \leq \tau_m$. Assume that

$$f(x^*, \ldots, x^*, \alpha_0) = 0.$$

Then x^* is a stationary point at $\alpha = \alpha_0$ and to which we can associate the linearized problem

$$\dot{y}(t) = A_0(x^*, \alpha_0)y(t) + \sum_{i=1}^{m} A_i(x^*, \alpha_0)y(t - \tau_i), \tag{2}$$

where, using $f \equiv f(x^0, x^1, \ldots, x^m, \alpha)$,

$$A_i(x^*, \alpha_0) = \frac{\partial f}{\partial x^i}(x^*(0), x^*(-\tau_1), \ldots, x^*(-\tau_m), \alpha_0),\ i = 0, \ldots, m$$

Define the $n \times n$-dimensional matrix Δ as

$$\Delta(x^*, \alpha_0, \lambda) = \lambda I - A_0(x^*, \alpha_0) - \sum_{i=1}^{m} A_i(x^*, \alpha_0)e^{-\lambda\tau_i}. \tag{3}$$

Then the characteristic equation reads

$$\det(\Delta(x^*, \alpha_0, \lambda)) = 0. \tag{4}$$

We refer to roots of the characteristic equation as eigenvalues. The stability of a generic stationary point is determined by its eigenvalues, see [Diekmann et al., 2012]. In particular, when eigenvalues cross the imaginary axis in the complex plane, bifurcations occur. To simplify notation we write $\Delta(\lambda)$ for $\Delta(x^*, \alpha_0, \lambda)$ when no confusion arises.

3 Detecting and locating codimension-1 bifurcation

3.1 Test functions

A function along the considered branch that has a regular zero at a bifurcation is called a test function. Below we will simply list functions used for detecting certain bifurcations without proving their regularity.

There is a real number ξ such that all zeros of $\det\Delta(\lambda)$ are in the left half plane $\{z \in \mathbb{C} | \mathrm{Re} z < \xi\}$. If in the linearized problem (2) not all A_i vanish simultaneously, then $\det\Delta(\lambda)$ has infinitely many roots such that there is only a finite number of solutions in any vertical strip in the complex plane. For detecting bifurcations depending on the eigenvalues we can therefore look at a vertical strip containing zero. Care must be taken when constructing test functions based on these eigenvalues in a chosen vertical strip. Indeed, the number of eigenvalues in the vertical strip may vary as parameters change. For this reason we prefer to use detection methods relaying on the Jordan chains of the characteristic matrix $\Delta(\lambda)$ whenever possible.

After a test function indicates a bifurcation by changing its sign, we need to locate the bifurcation point accurately in order to calculate the critical normal form coefficients. As we will see below, we use the bisection method or apply Newton on a specially constructed defining system to achieve this.

When continuing a steady-state of a DDE, we generally expect to encounter codimension-1 bifurcations, that is fold or Hopf bifurcations. We therefore start by defining test functions for these two bifurcations.

3.2 Fold ($\lambda_1 = 0$)

At a fold point the characteristic equation $\det\Delta(\lambda) = 0$ has a simple eigenvalue $\lambda_1 = 0$ and no other eigenvalues on the imaginary axis. A test function to detect this bifurcation is given by

$$\psi_f = \det\Delta(0).$$

To locate the fold point, the defining system

$$S_f(x, q, \alpha) = \begin{pmatrix} f(x, \ldots, x, \alpha) \\ \Delta(x, \alpha, 0)q \\ c^T q - 1 \end{pmatrix} = 0 \tag{5}$$

is used. Here $c^T q - 1$ presents a suitable normalization of $q \in \mathbb{R}^n$. The vector $c \in \mathbb{R}^n$ is chosen as $c = q^{(0)}/(q^{(0)^T} q^{(0)})$, where $q^{(0)}$ is the initial value of q. Solving this system by Newton's method gives (x, q, α), corresponding to the fold point.

3.3 Hopf ($\lambda_{1,2} = \pm i\omega_0$)

At a Hopf point the characteristic equation $\det\Delta(\lambda) = 0$ has a pair of purely complex conjugate eigenvalues $\lambda = \pm i\omega$, with $\omega > 0$, and no other eigenvalues on the imaginary axis. For a test function we look at the sign of the smallest real part of the complex eigenvalue pairs. For this the following function has been constructed

```
function [smallest_real_part, stability,selectedroot] = nmfm_smrp(...
funcs,point, stmethod,varargin)
%% Compute smallest real part of imaginary pairs or of real eigenvalues
%
% * rmomega (logical): remove known imaginary roots
% * threshold (function): threshold(abs(imag(roots)) selects roots to
% consider
%
% $Id: nmfm_smrp.m 73 2014-12-31 10:47:51Z jan.sieber $
%
%% default={'stabilityfield','l1','remove_omega',false, ...
%  'threshold',@(x)true(size(x))};
options=dde_set_options(default,varargin);
if ~isfield(point, 'stability') || isempty(point.stability) || ...
                isempty(point.stability.l0)
        point.stability = p_stabil(funcs,point, stmethod);
end

stability = point.stability;
roots = stability.(options.stabilityfield);

%% Remove roots closest to known eigenvalue pair
if options.remove_omega
   [dum,ix]=sort(abs(roots - 1i*point.omega)); %#ok<ASGLU>
   roots(ix(1))=[];
   [dum,ix]=sort(abs(roots + 1i*point.omega)); %#ok<ASGLU>
   roots(ix(1))=[];
end
```

```
selectedroots = roots(options.threshold(abs(imag(roots))));
if isempty(selectedroots)
   smallest_real_part = 0;
   return
end
%% Assume all imaginary eigenvalues come in pairs
realparts = real(selectedroots);
[dum, rpind] = sort(abs(realparts)); %#ok<ASGLU>
smallest_real_part = realparts(rpind(1));
selectedroot=selectedroots(rpind(1));
end
```

The value of the test function ψ_H at the ith point on the steady-state branch is then given by

$$\psi_H(i) = \text{nmfm\_smrp(funcs,p,stmethod,'remove\_omega',false,'threshold',isimag).}$$

Here `isimag=@(x)x>imagthresh` is an anonymous function that selects all eigenvalues that have imaginary parts greater than `imagthresh`. If there are no complex eigenvalues according to this rule the function returns the zero value. Since a sign change is monitored by

$$|\text{sign}(\phi_H(i)) + \text{sign}(\phi_H(i-1))|,$$

we have to exclude the case $\phi_H(i) = 0$ while detecting.

When a bifurcation is detected we locate the Hopf point using the defining system

$$S_H(x, q, \alpha, \omega) = \begin{pmatrix} f(x, \ldots, x, \alpha) \\ \Delta(x, \alpha, i\omega)q \\ c^H q - 1 \end{pmatrix} = 0. \tag{6}$$

Here $c^H q - 1$ presents a suitable normalization of $q \in \mathbb{C}^n$. The vector $c \in \mathbb{C}^n$ is chosen as $c = q^{(0)}/(q^{(0)^H} q^{(0)})$, where $q^{(0)}$ is the initial value of q. Solving this system by Newton's method gives (x, q, α, ω), corresponding to the Hopf point.

4 Detecting and locating codimension-2 bifurcations

Due to the existence of a finite-dimensional smooth center manifold for Delay Differential Equations it is possible to 'lift' the normal form theory for Ordinary Differential Equations (ODEs) to DDEs. The ODE on the center manifold can be transformed into a normal form by smooth invertible substitutions of the coordinates. The critical normal form coefficients can be calculated with different methods. In [Janssens, 2007] the theory of sun-star calculus was combined with the approach used in [Kuznetsov, 2004] for calculating critical normal form for ODEs. We refer to [Janssens, 2007] for the derivation of the critical normal form coefficients. Here we will simply list the critical normal forms on the center manifold and demonstrate how to compute their coefficients. Readers familiar with normal forms for ODEs will notice the similarity. The major difference is seen in the vectors used in the critical coefficients. In ODEs eigenvectors and generalized eigenvectors of the Jacobian of the system appear in the coefficients. In DDEs the vectors used in the coefficients are obtained from the Jordan chains of the characteristic matrix (3).

When investigating two-parameter problems, one usually encounters higher-order degeneracies along codimension-1 bifurcation curves. Some of these degeneracies are determined by the characteristic matrix, while others can only be detected using the non-linear terms of (1). For this reason we start the section with the critical normal forms for codimension-1 equilibrium bifurcations, namely the fold and Hopf.

4.1 Normal forms for codimension-1 bifurcations

Consider the steady-state x^* of (1) at $\alpha = \alpha_0$. By a shift of coordinates and parameters it can always be arranged that $x^* = 0$ and $\alpha_0 = 0$. Since we are only calculating the critical normal form coefficients, we drop the parameter and write $f(x^0, x^1, \ldots, x^m)$. To simplify notation we write f^0 for $f(x^*, x^*, \ldots, x^*)$. Expanding f in $X = \left(x^0, x^1, \ldots, x^m\right) \in \mathbb{R}^n \times \mathbb{R}^{m+1}$ around $(x^*, x^*, \ldots, x^*)$ yields

$$\begin{aligned} f(X) = \left(Df^0\right) X + \frac{1}{2}\left(D^2 f^0\right)(X,X) + \frac{1}{3!}\left(D^3 f^0\right)(X,X,X) \\ + \frac{1}{4!}\left(D^4 f^0\right)(X,X,X,X) + \frac{1}{5!}\left(D^5 f^0\right)(X,X,X,X,X) + \mathcal{O}(X^6), \end{aligned} \tag{7}$$

where

$$\begin{aligned} Df_i^0(q) &= \sum_{j=0}^{m} \frac{\partial f_i^0}{\partial x^j} q^j = \sum_{k=1}^{n}\sum_{j=0}^{m} \frac{\partial f_i^0}{\partial x_k^j} q_i^j, \\ D^2 f_i^0(q,p) &= \sum_{j_1,j_2=0}^{m} \frac{\partial f_i^0}{\partial x^{j_1}\partial x^{j_2}} q^{j_1} p^{j_2} \\ &= \sum_{k_1,k_2=1}^{n}\sum_{j_1,j_2=0}^{m} \frac{\partial f_i^0}{\partial x_{k_1}^{j_1}\partial x_{k_2}^{j_2}} q_{i_1}^{j_1} p_{i_2}^{j_2}, \end{aligned}$$

for $1 \le i \le n$. The multilinear forms $D^3 f^0(p,q,z)$, $D^4 f^0(p,q,z,v)$ and $D^5 f^0(p,q,z,v,w)$ can be expressed similarly.

4.1.1 Fold ($\lambda_1 = 0$, $b \neq 0$)

If the steady-state $x^* \equiv 0$ of (1) has a fold bifurcation at the parameter value $\alpha_0 = 0$, then the characteristic equation $\det \Delta(\lambda) = 0$ has a simple zero $\lambda_1 = 0$. Then there exist vectors $q, p \in \mathbb{R}^n$ such that

$$\Delta(0)q = 0, \qquad p^T \Delta(0) = 0.$$

These can be normalized to satisfy

$$p^T \Delta'(0) q = 1$$

Here Δ' denotes the derivative of the characteristic equation (4) with respect to λ. The restriction of (1) to the one-dimensional center manifold W^c can be transformed to the form

$$\dot{w} = bw^2 + \mathcal{O}\left(|w|^3\right) \qquad w \in \mathbb{R}, \tag{8}$$

where the critical normal form coefficient is given by

$$b = \frac{1}{2} p^T D^2 f^0(\Phi, \Phi), \tag{9}$$

with

$$\Phi = (q, q, \ldots, q) \in \mathbb{R}^n \times \mathbb{R}^{m+1}. \tag{10}$$

When a fold point is detected and located, the critical normal form coefficient b is reported in the Matlab console.

4.1.2 Hopf ($\lambda_{1,2} = \pm i\omega_0$, $\ell_1 \neq 0$)

If the steady-state $x^* \equiv 0$ of (1) has a Hopf bifurcation at the parameter value $\alpha_0 = 0$, then the characteristic equation $\det \Delta(\lambda) = 0$ has a pair of purely imaginary eigenvalues $\lambda_{1,2} = \pm i\omega_0$, with $\omega_0 > 0$, and no other eigenvalues on the imaginary axis. Let $q, p \in \mathbb{R}^n$ such that

$$\Delta(i\omega_0)q = 0, \qquad p^T \Delta(i\omega_0) = 0.$$

These can be normalized to satisfy

$$p^T \Delta'(i\omega_0) q = 1.$$

The restriction of (1) to the two-dimensional center manifold W^c can be transformed into the form

$$\dot{z} = i\omega_0 z + c_1 z^2 \bar{z} + \mathcal{O}(|z|^4) \qquad z \in \mathbb{C}.$$

The critical normal form coefficient in this expression is c_1. The first Lyapunov coefficient is given by

$$\ell_1 = \frac{1}{\omega_0} \operatorname{Re} c_1, \tag{11}$$

where

$$c_1 = \frac{1}{2} p^T \left[D^2 f^0\left(\bar{\Phi}, H_{20}\right) + 2 D^2 f^0\left(\Phi, H_{11}\right) + D^3 f^0(\Phi, \Phi, \overline{\Phi}) \right],$$

with Φ, H_{20} and H_{11} computed as follows. Let

$$\begin{aligned} h_{20}(\vartheta) &= e^{2i\omega_0\vartheta} \Delta(\phi, \alpha_0, 2i\omega_0)^{-1} D^2 f^0(\Phi, \Phi), \\ h_{11}(\vartheta) &= \Delta(\phi, \alpha_0, 0)^{-1} D^2 f^0(\Phi, \overline{\Phi}), \\ \phi(\vartheta) &= e^{i\omega_0\vartheta} q, \end{aligned} \tag{12}$$

then

$$\begin{aligned} \Phi &= \left(\phi(0), \phi(-\tau_1), \ldots, \phi(-\tau_m)\right), \\ H_{20} &= \left(h_{20}(0), h_{20}(-\tau_1), \ldots, h_{20}(-\tau_m)\right), \\ H_{11} &= \left(h_{11}(0), h_{11}(-\tau_1), \ldots, h_{11}(-\tau_m)\right). \end{aligned} \tag{13}$$

When a Hopf point is detected and located, the critical normal form coefficient ℓ_1 is reported in the Matlab console.

4.2 Codimension-2 points on a fold curve $(\lambda_1 = 0)$

While continuing a fold branch, the following codim-2 points can be encountered:

1. An additional real eigenvalue λ_2 meets the imaginary axis, with their geometric multiplicity remaining one, while the center manifold W^c becomes two-dimensional:

   $$\lambda_{1,2} = 0.$$

   These are the conditions for a Bogdanov-Takens bifurcation. A test function to detect this bifurcation is given by

   $$\psi_{BT}^f = p^T \Delta'(0) q.$$

   Here the vectors p, q are calculated using the function `eig` from `Matlab`. Depending on the system being investigated, this function does not always pass zero when a Bogdanov-Takens bifurcation takes place on a Fold curve. For example the system in Section (9) exhibits this behavior, see Figure 15. Therefore we also monitor the derivative of ψ_{BT}^f with respect to the parameters. We note that at a Bogdanov-Takens point the normalization for the calculation of the critical normal form coefficient b of a fold point cannot be achieved anymore. This we also see in Figure 15.

2. Two extra non-real eigenvalues $\lambda_{2,3}$ meet the imaginary axis, and W^c becomes three-dimensional:

   $$\lambda_1 = 0, \qquad \lambda_{2,3} = \pm i\omega_0,$$

   for $\omega_0 > 0$. These conditions correspond to the fold-Hopf bifurcation, also known as the Gavrilov-Guckenheimer bifurcation. The test function is the same as the test function for a Hopf bifurcation on a steady-sate branch, see Section (3.3)

3. The eigenvalue $\lambda_1 = 0$ remains simple and the only one on the imaginary axis (dim $W^c = 1$), but the normal form coefficient a in equation (8) vanishes

   $$\lambda_1 = 0,\ a = 0.$$

   These are the conditions for a cusp bifurcation. The coefficient b given in (9) can be used as a test function to detect this bifurcation:

   $$\psi_{CP}^f = b,$$

   where b is given by (9).

4.3 Codimension-2 points on a Hopf curve $(\lambda_{1,2} = \pm i\omega_0)$

While continuing a Hopf branch, the following codim-2 points can be encountered:

1. An additional eigenvalue $\lambda_3 = 0$ meets the imaginary axis and W^c becomes three-dimensional:

   $$\lambda_{1,2} = \pm i\omega_0, \qquad \lambda_3 = 0.$$

   As seen before these are the conditions for a fold-Hopf bifurcation. For detection we can use the test function for a fold

   $$\psi_{FH}^H = \det\left(\Delta(0)\right). \tag{14}$$

Branch	Bifurcation	Test function
steady-state	fold	$\det \Delta(0)$
	Hopf	smallest real part of the complex eigenvalue pairs
fold	cusp	critical normal form coefficient a
	Bogdanov-Takens	$\psi_{BT}^f = p_1^T \Delta'(0) p_0$ or the derivative of ψ_{BT}^f
	fold-Hopf	smallest real part of the complex eigenvalue pairs
Hopf	Bogdanov-Takens	ω_0
	fold-Hopf	$\det \Delta(0)$
	double-Hopf	smallest real part of the complex eigenvalue pairs
	generalized Hopf	first Lyapunov coefficient ℓ_1

Table 1: An overview of bifurcations and monitored test functions.

2. A Bogdanov-Takens bifurcation occurs when the two purely imaginary eigenvalues λ_1 and λ_2 collide. We use
$$\psi_{BT}^H = \omega_0$$
as a test function.

3. An additional pair of non-real eigenvalues $\lambda_{3,4} = \pm i\omega_1$, meet the imaginary axis and W^c becomes four-dimensional:
$$\lambda_{1,2} = \pm i\omega_0, \qquad \lambda_{3,4} = \pm i\omega_1.$$
These conditions indicate a double-Hopf bifurcation. The value of the test function for a double-Hopf bifurcation for the ith point on a Hopf branch is given by
$$\psi_H(i) = \text{nmfm\_mrp(funcs,p,stmethod,'remove\_omega',true,'threshold',isimag).}$$

4. As in the fold case, where $\lambda_1 = 0$ remains a simple zero and the bifurcation is determined by a critical normal form coefficient, there is also a bifurcation in the Hopf case where the dimension W^c remains the same. When the first Lyapunov coefficient ℓ_1 from (11) vanishes we have a generalized Hopf bifurcation. The test function for a generalized Hopf point is
$$\psi_{GH} = \ell_1.$$

5 Implementation in DDE-BifTool

The call to the bifurcation routine is as follows:

```
[newbranch, success] = br_bifdet(branch)
```

Here, `branch` is either a steady-state branch, fold branch or Hopf branch, for which the stability has been calculated with `br_stabl`. The `newbranch` is a copy of `branch`, but with possible

bifurcation points added. If bifurcation points are added `success` is set to 1, otherwise 0 will be returned. The function `br_bifdet` will calculate the test function from Section (3.1), (4.2) or (4.3), depending on type of branch, for every point on the branch. In Table 1 these test functions are summarized. If a sign change occurs the point is located. For fold and Hopf points the function `p_correct`, from the default package, is used. For codim-2 bifurcations the auxiliary function `locate_codim2` has been written. After a bifurcation point is located successfully the normal form coefficients are computed and stored in the `nmfm` structure of the corrected point. Lastly the function `add_to_branch` is called to add the corrected bifurcation point to the `newbranch`. In this function is checked if the distance between the bifurcation point and the line connecting the last and second to last points on the branch is no more than 25% of the length of this line. If this condition is satisfied the corrected bifurcation point is inserted into `newbranch`.

5.1 Fields structure of `method.bifurcation`

Fields of the structure `method.bifurcation` , which is a substructure of a branch structure.

- `correction_tolerance`: Custom value for `method.point.minimal_accuracy` during codimension-1 bifurcations correction. The default value is 10^{-7}.
- `radial_tolerance_factor`: The maximum distance to the branch the bifurcation point may have, relative to the distance between the last two points. The default value is 0.25.
- `secant_iterations`: The maximum number of iterations for the bisection method used to correct codimension-2 points. The default value is 30.
- `secant_tolerance`: The minimal accuracy for the bisection method. The default value is 10^{-9}.
- `plot_testfunctions`: When set to 1 the test functions are plotted. The default value is 0.
- `monitor_eigenvalues`: When set to 1 the eigenvalues of the ith point on the branch are plotted. The default value is 0.
- `pause_on_bifurcation`: When set to 1 the script will pause when a bifurcation is detected and corrected. By pressing any key the script will continue. The default value is 0.
- `monitor_eigenvalues_time`: The time in seconds to pause after plotting the eigenvalues of the ith point on the branch. The default value is 0.1.
- `imagthreshold`: Threshold for treating a number as complex used in the function `nmfm_smrp`, see subsection 3.3. The default value is set to 10^{-9}. We note that setting this parameter to zero didn't seem to affect the examples used in this manual.

5.2 Locating codim-2 bifurcations, except Bogdanov-Takens

If a codimension-2 bifurcation is detected, we want to locate the bifurcation point. This is done with the bisection method:

1. Start with one point at which a test function is negative and one point at which the test function is positive

2. Construct the point halfway between the positive and the negative point
3. Correct this point
4. Compute the test function at this point
5. If the test function is negative, make this point the new negative point; otherwise, the new positive point
6. Repeat until the absolute value of the test function is smaller than some predefined tolerance

5.3 Locating Bogdanov-Takens

For a Bogdanov-Takens bifurcation the defining system

$$S(x,v,w,\alpha) = \begin{pmatrix} f(x,x,\ldots,x,\alpha) \\ \Delta(x,\alpha,0)v \\ \Delta'(x,\alpha,0)v + \Delta(x,\alpha,0)w \\ (v,v)-1 \\ (w,v) \end{pmatrix}$$

is used.

5.4 Flagging bifurcation points

When one has a branch of steady-states on which a Hopf bifurcation occurs, one typically wants to plot the branch as a line and give the Hopf point a special marker (e.g. a big dot). Ideally, one simply passes an entire branch to a plot routine, along with an instruction how to display the various bifurcations occurring along a branch.

Within the architecture of Matlab, this task provided a challenge. As DDE-BifTool branches are stored as simple arrays of point-structures, it was not possible to have a branch of `stst` points containing a `hopf` point as well, because different types of structures cannot occur in the same array. Therefore, we chose to introduce an extra field to all point structures, called `flag`, to store information on what kind of bifurcation occurs at this point. For example, a `stst` can have `stst.flag = 'hopf'` or a `hopf` can have hopf.flag = 'GH' (for generalized Hopf).

To make this work, two changes had to be made throughout all DDE-BifTool source files. First, all subroutines creating new point structures have been modified to also set `point.flag = ''` (so that all points in a branch have precisely the same structure). Second, all subroutines adding points to branches had to be modified at the lines where the concatenation takes place.

Given a branch with bifurcations, one also wants to extract all points flagged as a particular bifurcation. This is possible using the new function `br_getflags`:

FPI=br_getflags(branch) This function extracts a list of flagged point indices out of branch. `FPI(1,:)` holds the indices of all points marked as `hopf` , `FPI(2,:)` those of `fold` , etc.

To streamline this process, we set up an indexing scheme to uniquely identify a type of bifurcation by a number. Two new functions are available to simplify this process.

Index	Flag	Point type	Codimension
0	stst	staedy-state	0
1	hopf	Hopf	1
2	fold	Fold	1
3	psol	Periodic Solution	0
4	hcli	Homoclinic	1
5	GH	Generalized Hopf	2
6	HH	Double-Hopf	2
7	ZH	Fold-Hopf (zero-Hopf)	2
8	BT	Bogdanov-Takens	2
9	CP	Cusp	2

Table 2: Bifurcation type numbering scheme.

Point type	Fields	Reported in Matlab console
Fold	b	b
Hopf	L1	L1
Cusp	c	c
Bogdanov-Takens	a2,b2	a,b
Generalized Hopf	L2	L2
Fold-Hopf	g200, g110, g011 , g300, g111, g210 , g021, a, b, c, d, e, s, theta	s, theta
Double-Hopf	g2100, g1011, g1110, g0021, theta, delta	theta, delta and eigenvalues

Table 3: A list of normal form coefficients as stored in the `point.nmfm` structure and what is being reported in the Matlab console.

num=bif2num(bifstring) This function extracts a list of flagged point indices out of branch. FPI(1,:) holds the indices of all points marked as hopf , FPI(2,:) those of fold , etc.

bifstring=num2bif(num) Converts the bifurcation index num to its bifurcation type.

So, for instance, we have `num2bif(1) = 'hopf'` and `bif2num('fold') = 2`. See Table 2 for the full specification.

5.5 Storage of the normal form coefficients

As was already indicated above, the critical normal form coefficients are stored in the new point field structure `nmfm` . This field contains all normal form coefficients as elements. For the specifics, see Table 3

All point manipulation routines have been updated to ensure that the `nmfm` field is always present in all point types. Note, however, that not all point manipulation routines actually compute the normal form coefficients when creating a new point: they leave the field empty. We chose to do this in order to reduce overhead in contexts where the normal form coefficients

are not specifically wanted. The only place where the normal form coefficients are set is in `br_contn` if bifurcation detection is enabled.

6 Critical normal forms for codimension-2 bifurcations

In some cases of the calculations of the critical normal form coefficients we use implicitly defined solutions or certain linear systems. Suppose that $\lambda \in \mathbb{C}$ is a simple root of $\Delta(\lambda)$. Let q be a null vector of $\Delta(\lambda)$ and p be a null vector of the transpose matrix $\Delta(\lambda)^T$. Then the *bordered inverse*

$$\Delta(\lambda)^{INV} y$$

is the unique solution x of the nonsingular linear system

$$\begin{pmatrix} \Delta(\lambda) & q \\ p^T & 0 \end{pmatrix} \begin{pmatrix} x \\ s \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}.$$

Furthermore the operator B_λ^{INV} is defined by

$$B_\lambda^{INV}(\zeta, \kappa) = \left(\theta \mapsto e^{\lambda\theta} \left(\Delta(\lambda)^{INV} \left(\zeta + \kappa \Delta'(\lambda) q \right) + \left(-p\Delta'(\lambda)\xi + \frac{1}{2} p\Delta''(\lambda) q \right) q - \kappa\theta q \right) \right)$$

for $(\theta \in [-h, 0])$.

6.1 Cusp $(\lambda = 0,\ a = 0,\ c \neq 0)$

If the steady-state $x^* \equiv 0$ of (1) has a cusp bifurcation at the parameter value $\alpha_0 = 0$, then the characteristic equation $\det \Delta(\lambda) = 0$ has a simple zero $\lambda_1 = 0$ and the critical normal form coefficient $b = 0$ in (9). Let $q, p \in \mathbb{R}^n$ be as in the fold case. The restriction of (1) to the one-dimensional center manifold W^c has the form

$$\dot{w} = cw^3 + \mathcal{O}\left(|w|^4\right) \qquad w \in \mathbb{R}. \tag{15}$$

The critical normal form coefficient is given by

$$c = \frac{1}{6} p^T \left[3D^2 f^0(\Phi, H_2) + D^3 f^0(\Phi, \Phi, \Phi) \right], \tag{16}$$

Where H_2 is computed as follows. Let

$$h_2 = -\Delta(0)^{INV} D^2 f^0(\Phi, \Phi) + p^T \Delta'(0) \Delta(0)^{INV} D^2 f^0(\Phi, \Phi) q \tag{17}$$

then

$$H_2 = (h_2, h_2, \ldots, h_2) \in \mathbb{R}^n \times \mathbb{R}^{m+1}$$

and Φ is as in (10). When a cusp point is detected and located, the critical normal form coefficients b from (9) and c from (16) are reported in the Matlab console.

6.2 Bogdanov-Takens $(\lambda_1 = \lambda_2 = 0)$

If the steady-state $x^* \equiv 0$ of (1) has a Bogdanov-Takens bifurcation at the parameter value $\alpha_0 = 0$, then there are vectors $q_0, q_1, p_1, p_0 \in \mathbb{R}^n$ such that the following relations hold

$$\begin{aligned} \Delta(0)q_0 &= 0, \\ \Delta'(0)q_0 &= -\Delta(0)q_1, \\ p_1^T\Delta(0) &= 0, \\ p_1^T\Delta(0)' &= -p_0^T\Delta(0). \end{aligned} \tag{18}$$

These can be normalized to satisfy

$$\begin{aligned} 1 &= p_0^T\Delta(0)'q_0 + \frac{1}{2!}p_1^T\Delta(0)''q_0, \\ 0 &= p_0^T\Delta(0)'q_1 + \frac{1}{2!}p_0^T\Delta(0)''q_0 + \frac{1}{2!}p_1^T\Delta(0)''q_1 + \frac{1}{3!}p_1^T\Delta(0)'''q_0. \end{aligned} \tag{19}$$

The restriction of (1) to the two-dimensional center manifold W^c at the critical parameter values can be transformed to the normal form

$$\begin{cases} \dot{w}_0 &= w_1, \\ \dot{w}_1 &= aw_0^2 + bw_0w_1 + \mathcal{O}(\| (w_0, w_1) \|^3), \end{cases}$$

where $a, b, z_0, z_1 \in \mathbb{R}$. Define the functions

$$\begin{aligned} \phi_0(\vartheta) &= q_0, \\ \phi_1(\vartheta) &= \vartheta q_0 + q_1, \end{aligned}$$

then the critical normal form coefficients are given by

$$\begin{aligned} a &= \frac{1}{2}p_1^T D^2 f^0 \left(\Phi_0, \Phi_0\right), \\ b &= p_0^T D^2 f^0 \left(\Phi_0, \Phi_0\right) + p_1^T D^2 f^0 \left(\Phi_0, \Phi_1\right), \end{aligned} \tag{20}$$

where

$$\begin{aligned} \Phi_0 &= \left(\phi_0(0), \phi_0(-\tau_1), \ldots, \phi_0(-\tau_m)\right), \\ \Phi_1 &= \left(\phi_1(0), \phi_1(-\tau_1), \ldots, \phi_1(-\tau_m)\right). \end{aligned}$$

When a Bogdanov-Takens point is detected and located, the critical normal form coefficients a and b are reported in the Matlab console.

6.3 Generalized Hopf $(\lambda_{1,2} = \pm i\omega_0,\ \ell_1 = 0,\ \ell_2 \neq 0)$

If the steady-state $x^* \equiv 0$ of (1) has a Hopf bifurcation at the parameter value $\alpha_0 = 0$, then the characteristic equation $\det \Delta(\lambda_1) = 0$ has a simple pair of purely imaginary conjugate

eigenvalues $\lambda = \pm i\omega_0$ and all other eigenvalues are in the left or right open half plane. Let $q, p \in \mathbb{R}^n$ such that

$$\Delta(i\omega_0)q = 0, \qquad p^T\Delta(i\omega_0) = 0, \qquad p^T\Delta'(i\omega_0)q = 1.$$

Furthermore, suppose that the first Lyapunov coefficient $\ell_1 = 0$, then the restriction of (1) to the two-dimensional center manifold W^c can be transformed to the normal form

$$\dot{z} = i\omega_0 z + c_1 z^2\bar{z} + c_2 z^3\bar{z}^2 + \mathcal{O}(|z|^4),$$

where $c_1, c_2, z \in \mathbb{C}$ with $\operatorname{Re} c_1 = 0$. The second Lyapunov coefficient is defined to be

$$\ell_2 = \frac{1}{\omega_0}\operatorname{Re} c_2, \tag{21}$$

where

$$\begin{aligned}
c_2 = \frac{1}{12}p^T \big[&6D^2f^0(H_{11}, H_{21}) + 3D^2f^0(\overline{H}_{21}, H_{20}) \\
&+ 3D^2f^0(\overline{H}_{20}, H_{30}) + 3D^2f^0(\Phi, H_{22}) + 2D^2f^0(\overline{\Phi}, H_{31}) \\
&+ 6D^3f^0(\overline{\Phi}, H_{20}, H_{11}) + 6D^3f^0(\Phi, H_{11}, H_{11}) \\
&+ 3D^3f^0(\Phi, H_{20}, \overline{H}_{20}) + 6D^3f^0(\Phi, \overline{\Phi}, H_{21}) + 3D^3f^0(\Phi, \Phi, \overline{H}_{21}) \\
&+ D^3f^0(\overline{\Phi}, \overline{\Phi}, H_{30}) + 6D^4f^0(\Phi, \Phi, \overline{\Phi}, H_{11}) \\
&+ 3D^4f^0(\Phi, \overline{\Phi}, \overline{\Phi}, H_{20}) + D^4f^0(\Phi, \Phi, \Phi, \overline{H}_{20}) \\
&+ D^5f^0(\Phi, \Phi, \Phi, \overline{\Phi}, \overline{\Phi})\big].
\end{aligned}$$

Here the quantities involved are computed as follows. Let

$$\begin{aligned}
h_{20}(\vartheta) &= e^{2i\omega_0\vartheta}(\Delta(2i\omega_0))^{-1}D^2f^0[\Phi, \Phi], \\
h_{11}(\vartheta) &= (\Delta(0))^{-1}D^2f^0[\Phi, \overline{\Phi}], \\
h_{30}(\vartheta) &= e^{3i\omega_0\vartheta}(\Delta(3i\omega_0))^{-1}[3D^2f^0[\Phi, H_{20}] + D^3f^0[\Phi, \Phi, \Phi]], \\
c_1 &= \frac{1}{2}p^T \cdot \left[D^2f^0(\Phi, H_{20}) + 2D^2f^0(\Phi, H_{11}) + D^3f^0(\Phi, \Phi, \overline{\Phi})\right], \\
h_{21}(\vartheta) &= e^{\lambda\vartheta}c_1\left(2\theta - p\Delta''(i\omega_0)\right)q, \\
h_{31}(\vartheta) &= e^{2i\omega_0\vartheta}(\Delta(2i\omega_0))^{-1}\left[D^2f^0[\overline{\Phi}, H_{30}] + 3D^2f^0[H_{20}, H_{11}] + 3D^2f^0[\Phi, H_{21}]\right. \\
&\quad \left. + 3D^3f^0[\Phi, \overline{\Phi}, H_{20}] + 3D^3f^0[\Phi, \Phi, H_{11}] + D^4f^0[\Phi, \Phi, \Phi, \overline{\Phi}]\right] \\
&\quad - 6c_1(\Delta(2i\omega_0))^{-1}[\Delta'(2i\omega_0) - I - \theta\Delta(2i\omega_0)]H_{20}, \\
h_{22}(\vartheta) &= \Delta(0)^{-1}\left[2D^2f^0[\overline{\Phi}, H_{21}] + 2D^2f^0[H_{11}, H_{11}] + 2D^2f^0[\Phi, \overline{H_{21}}]\right. \\
&\quad + D^2f^0[H_{20}, \overline{H_{20}}] + D^3f^0[\overline{\Phi}, \overline{\Phi}, H_{20}] + D^3f^0[\Phi, \Phi, \overline{H_{20}}] \\
&\quad \left. + 4D^3f^0[\Phi, \overline{\Phi}, H_{11}] + D^4f^0[\Phi, \Phi, \overline{\Phi}, \overline{\Phi}]\right], \\
\phi(\vartheta) &= e^{i\omega_0\vartheta}q,
\end{aligned}$$

then

$$\begin{aligned}
\Phi &= (\phi(0), \phi(-\tau_1), \ldots, \phi(-\tau_m)), \\
H_{ij} &= (h_{ij}(-\tau_0), \ldots, h_{ij}(-\tau_m)),
\end{aligned}$$

for $(i, j) \in \{(2,0), (1,1), (3,0), (2,1), (3,1), (2,2)\}$. When a generalized Hopf point is detected and located, the critical normal form coefficient ℓ_2 from (21) is reported in the Matlab console.

6.4 Fold-Hopf ($\lambda_1 = 0$, $\lambda_{2,3} = \pm i\omega_0$)

If the steady-state $x^* \equiv 0$ of (1) has a fold-Hopf bifurcation at the parameter value $\alpha_0 = 0$, then there are eigenvectors $q_0 \in \mathbb{R}^n$ and $q_1 \in \mathbb{C}^n$, satisfying

$$\Delta(0)q_0 = 0, \qquad \Delta(i\omega_0)q_1 = 0,$$

and adjoint eigenvectors $p_0 \in \mathbb{R}^n$ and $p_1 \in \mathbb{C}^n$, satisfying

$$p_0^T\Delta(0) = 0, \qquad p_1^T\Delta(i\omega_0) = 0.$$

These can be normalized to satisfy

$$p_0^T\Delta'(0)q_0 = 1 \qquad \text{and} \qquad p_1^T\Delta'(i\omega_0)q_1 = 1.$$

The restriction of (1) to the three-dimensional center manifold W^c can be transformed to the normal form

$$\begin{cases} \dot{w} &= g_{200}w^2 + g_{011}|z|^2 + g_{300}w^3 + g_{111}w|z|^2 + \mathcal{O}\left(\| (w,z,\bar{z}) \|^4\right), \\ \dot{z} &= i\omega_0 z + g_{110}wz + g_{210}w^2z + g_{021}z|z|^2 + \mathcal{O}\left(\| (w,z,\bar{z}) \|^4\right), \end{cases}$$

where $w \in \mathbb{R}$, $z \in \mathbb{C}$ and the coefficients g_{jkl} are real in the first equation and complex in the second. Suppose that the b and c are real constants such that $g_{110}g_{011} \neq 0$, then, generically, the restriction of (1) to the three-dimensional center manifold W^c can be reduced to the system

$$\begin{cases} \dot{w} &= bw^2 + c|z|^2 + \mathcal{O}(\| (w,z,\bar{z}) \|^4), \\ \dot{z} &= i\omega_0 z + dwz + ew^2z + \mathcal{O}(\| (w,z,\bar{z}) \|^4), \end{cases}$$

where

$$b = g_{200}, c = g_{011}, d = g_{110} - i\omega_0\frac{g_{300}}{g_{200}}$$

and

$$e = \mathrm{Re}\left(g_{210} + g_{110}\left(\frac{\mathrm{Re}g_{021}}{g_{011}} - \frac{3}{2}\frac{g_{300}}{g_{200}} + \frac{g_{111}}{2g_{011}}\right) - \frac{g_{021}g_{200}}{g_{011}}\right).$$

Depending on the signs of

$$s = bc \qquad \text{and} \qquad \theta = \mathrm{Re}\,\frac{g_{100}}{g_{200}}$$

4 bifurcation diagrams for nearby parameter values can be distinguished, see [Kuznetsov, 2004].

Define the functions

$$\begin{aligned} h_{200}(\vartheta) &= B_0^{INV}\left(D^2f^0(\Phi_0,\Phi_0), -p_0\cdot D^2f^0(\Phi_0,\Phi_0)\right), \\ h_{020}(\vartheta) &= e^{2i\omega_0\theta}\Delta(2i\omega_0)^{INV}D^2f^0(\Phi_1,\Phi_1), \\ h_{110}(\vartheta) &= B_{i\omega_0}^{INV}\left(D^2f^0(\Phi_0,\Phi_1), -p_1\cdot D^2f^0(\Phi_0,\Phi_1),\right), \\ h_{011}(\vartheta) &= B_0^{INV}\left(D^2f^0(\Phi_1,\overline{\Phi_1}), -p_0\cdot D^2f^0(\Phi_1,\overline{\Phi_1})\right), \\ \phi_0(\vartheta) &= q_0, \\ \phi_1(\vartheta) &= e^{i\omega_1\vartheta}q_1, \end{aligned}$$

then the critical normal form coefficients become

$$
\begin{aligned}
g_{200} &= \frac{1}{2} p_0^T D^2 f^0(\Phi_0, \Phi_0), \\
g_{110} &= p_1^T D^2 f^0(\Phi_0, \Phi_1), \\
g_{011} &= p_0^T D^2 f^0(\Phi_1, \overline{\Phi_1}), \\
g_{300} &= \frac{1}{6} p_0^T \left(3D^2 f^0(\Phi_0, H_{200}) + D^3 f^0(\Phi_0, \Phi_0, \Phi_0) \right), \\
g_{111} &= p_0^T \left(D^2 f^0(\Phi_0, H_{011}) + D^2 f^0(\overline{\Phi_1}, H_{110}) + D^2 f^0(\Phi_1, H_{110}) + D^3 f^0(\Phi_0, \Phi_0, \overline{\Phi_1}) \right), \\
g_{210} &= \frac{1}{2} p_1^T \left(D^2 f^0(\Phi_1, H_{200}) + 2D^2 f^0(\Phi_0, H_{110}) + D^3 f^0(\Phi_0, \Phi_0, \Phi_1) \right), \\
g_{021} &= \frac{1}{2} p_1^T \left(D^2 f^0(\overline{\Phi_1}, H_{020}) + 2D^2 f^0(\Phi_1, H_{011}) + D^3 f^0(\Phi_1, \Phi_1, \overline{\Phi_1}) \right),
\end{aligned}
$$

where

$$
\begin{aligned}
H_{jkl} &= (h_{jkl}(-\tau_0), \ldots, h_{jkl}(-\tau_m)) \in \mathbb{R}^n \times \mathbb{R}^{m+1}, \\
\Phi_0 &= (\phi_0(0), \phi_0(-\tau_1), \ldots, \phi_0(-\tau_m)), \\
\Phi_1 &= (\phi_1(0), \phi_1(-\tau_1), \ldots, \phi_1(-\tau_m)).
\end{aligned}
$$

When a fold-Hopf point is detected and located, the quantities s and θ are reported in the Matlab console.

6.5 Double-Hopf $(\lambda_{1,2} = \pm i\omega_1 \, \lambda_{1,2} = \pm i\omega_2)$

If the steady-state $x^* \equiv 0$ of (1) has a double-Hopf bifurcation at the parameter value $\alpha_0 = 0$, then there are eigenvectors $q_1, q_2 \in \mathbb{C}^n$, satisfying

$$
\Delta(i\omega_1) q_1 = 0, \qquad \Delta(i\omega_2) q_2 = 0,
$$

and adjoint eigenvectors $p_1, p_2 \in \mathbb{C}^n$, satisfying

$$
p_1^T \Delta(i\omega_1) = 0, \qquad p_2^T \Delta(i\omega_2) = 0.
$$

These can be normalized to satisfy

$$
p_1^T \Delta'(i\omega_1) q_1 = 1 \qquad \text{and} \qquad p_2^T \Delta'(i\omega_2) q_2 = 1.
$$

Assume that

$$
k\omega_1 \neq l\omega_2, \qquad k, l > 0, \qquad k + l \leq 5, \tag{22}
$$

where k, l are integer numbers. The restriction of (1) to the four-dimensional center manifold W^c can be transformed to the normal form

$$
\begin{cases}
\dot{z}_1 &= i\omega_1 z_1 + g_{2100} z_1 |z_1|^2 + g_{1011} z_1 |z_2|^2 + g_{3200} z_1 |z_1|^4 \\
&\quad + g_{2111} z_1 |z_1|^2 |z_2|^2 + g_{1022} z_1 |z_2|^4 + \mathcal{O}\left(\|z_1, \overline{z_1}, z_2, \overline{z_2}\|^6 \right), \\
\dot{z}_2 &= i\omega_2 z_2 + g_{1110} z_2 |z_1|^2 + g_{0021} z_2 |z_2|^2 + g_{2210} z_2 |z_1|^4 \\
&\quad + g_{1121} z_2 |z_1|^2 |z_2|^2 + g_{0032} z_2 |z_2|^4 + \mathcal{O}\left(\|z_1, \overline{z_1}, z_2, \overline{z_2}\|^6 \right),
\end{cases}
$$

where $z_1, z_2 \in \mathbb{C}^2$ and $g_{jklm} \in \mathbb{C}$. Moreover, if

$$(\operatorname{Re} g_{2100})\,(\operatorname{Re} g_{1011})\,(\operatorname{Re} g_{1110})\,(\operatorname{Re} g_{0021}) \neq 0$$

and the critical eigenpairs cross the imaginary axis with nonzero velocities, then the system (1) can be reduced to the system

$$\begin{cases} \dot{z}_1 &= i\omega_1 z_1 + \frac{1}{2}p_{11}z_1|z_1|^2 + p_{12}z_1|z_2|^2 + ir_1 z_1|z_1|^4 \\ &\quad + \frac{1}{4}s_1 z_1|z_2|^4 + \mathcal{O}\left(\|z_1, \overline{z_1}, z_2, \overline{z_2}\|^6\right), \\ \dot{z}_2 &= i\omega_2 z_2 + p_{21}z_2|z_1|^2 + \frac{1}{2}p_{22}z_2|z_2|^2 + \frac{1}{4}s_2 z_2|z_1|^4 \\ &\quad + ir_2 z_2|z_2|^4 + \mathcal{O}\left(\|z_1, \overline{z_1}, z_2, \overline{z_2}\|^6\right), \end{cases}$$

where the coefficients P_{jk} and S_k are complex, while the numbers R_k are real. Moreover, the real parts of the critical values are given by the expressions

$$\operatorname{Re}\begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix} = \operatorname{Re}\begin{pmatrix} g_{2100} & g_{1011} \\ g_{1110} & g_{0021} \end{pmatrix}$$

and

$$\operatorname{Re} s_1 = \operatorname{Re} g_{1022} + \frac{1}{3}\operatorname{Re} g_{1011}\left(6\frac{\operatorname{Re} g_{1121}}{\operatorname{Re} g_{1110}} - 4\frac{\operatorname{Re} g_{0032}}{\operatorname{Re} g_{0021}} - 6\frac{(\operatorname{Re} g_{3200})\,(\operatorname{Re} g_{0021})}{(\operatorname{Re} g_{2100})\,(\operatorname{Re} g_{1110})}\right),$$
$$\operatorname{Re} s_2 = \operatorname{Re} g_{2210} + \frac{1}{3}\operatorname{Re} g_{1110}\left(6\frac{\operatorname{Re} g_{2111}}{\operatorname{Re} g_{1011}} - 4\frac{\operatorname{Re} g_{3200}}{\operatorname{Re} g_{2100}} - 6\frac{(\operatorname{Re} g_{2100})\,(\operatorname{Re} g_{0032})}{(\operatorname{Re} g_{1011})\,(\operatorname{Re} g_{0021})}\right).$$

Depending on the sign of

$$(\operatorname{Re} p_{11})\,(\operatorname{Re} p_{22}) = (\operatorname{Re} g_{2100})\,(\operatorname{Re} g_{0021}),$$

this bifurcation exhibits either 'simple' or 'difficult' dynamics for nearby parameter values. Each case includes many subcases depending on the signs of

$$\theta = \frac{\operatorname{Re} g_{1011}}{\operatorname{Re} g_{0021}}, \qquad \delta = \frac{\operatorname{Re} g_{1110}}{\operatorname{Re} g_{21000}},$$

see [Kuznetsov, 2004]. Define the functions

$$\begin{aligned} h_{1100}(\vartheta) &= \Delta(0)^{-1} D^2 f^0(\Phi_1, \overline{\Phi}_1), \\ h_{2000}(\vartheta) &= e^{2i\omega_1\vartheta}\Delta(2i\omega_1)^{-1} D^2 f^0(\Phi_1, \Phi_1) \\ h_{1010}(\vartheta) &= e^{i(\omega_1+\omega_2)\vartheta}\Delta(i(\omega_1+\omega_2))^{-1} D^2 f^0(\Phi_1, \Phi_2), \\ h_{1001}(\vartheta) &= e^{i(\omega_1-\omega_2)\vartheta}\Delta(i(\omega_1-\omega_2))^{-1} D^2 f^0(\Phi_1, \overline{\Phi}_2), \\ h_{0020}(\vartheta) &= e^{2i\omega_2\vartheta}\Delta(2i\omega_2)^{-1} D^2 f^0(\Phi_2, \Phi_2), \\ h_{0011}(\vartheta) &= \Delta(0)^{-1} D^2 f^0(\Phi_2, \overline{\Phi}_2), \\ \phi_1(\vartheta) &= e^{i\omega_1\vartheta} q_1, \\ \phi_2(\vartheta) &= e^{i\omega_2\vartheta} q_2, \end{aligned}$$

then the cubic coefficients become

$$
\begin{aligned}
g_{2100} &= \frac{1}{2}p_1^T\left\langle 2D^2f(0)(H_{1100},\Phi_1)+D^2f(0)(H_{2000},\overline{\Phi}_1)+D^3f(0)(\Phi_1,\Phi_1,\overline{\Phi}_1)\right\rangle,\\
g_{1011} &= p_1^T\left\langle D^2f(0)(H_{0011},\Phi_1)+D^2f(0)(H_{1001},\Phi_2)\right.\\
&\qquad \left.+D^2f(0)(H_{1010},\overline{\Phi}_2)+D^3f(0)(\Phi_1,\Phi_2,\overline{\Phi}_2)\right\rangle,\\
g_{1110} &= p_2^T\left\langle D^2f(0)(\overline{H_{1001}},\Phi_1)+D^2f(0)(H_{1010},\overline{\Phi}_1)\right.\\
&\qquad \left.+D^2f(0)(H_{1100},\Phi_2)+D^3f(0)(\Phi_1,\overline{\Phi}_1,\Phi_2)\right\rangle,\\
g_{0021} &= \frac{1}{2}p_2^T\left\langle 2D^2f(0)(H_{0011},\Phi_2)+D^2f(0)(H_{0020},\overline{\Phi}_2)+D^3f(0)(\Phi_2,\Phi_2,\overline{\Phi}_2)\right\rangle,
\end{aligned}
$$

where

$$
\begin{aligned}
H_{jklm} &= (h_{jklm}(0),h_{jklm}(-\tau_1)\ldots,h_{jklm}(-\tau_m))\in\mathbb{R}^n\times\mathbb{R}^{m+1},\\
\Phi_1 &= (\phi_1(0),\phi_1(-\tau_1),\ldots,\phi_1(-\tau_m)),\\
\Phi_2 &= (\phi_2(0),\phi(-\tau_1),\ldots,\phi_2(-\tau_m)).
\end{aligned}
$$

Higher order coefficients are not implemented. When a double-Hopf point is detected and located the quantities θ and δ are reported in the Matlab console. To verify the non-resonance conditions from (22) the eigenvalues are also reported. In that case the normal form coefficients do not apply.

7 Example of a system with Hopf and degenerate Hopf bifurcations

The following non-dimensionalized model of two interacting layers of neurons is considered

$$
\begin{cases}
\dot{x}_1(t) &= -x_1(t)-ag(bx_1(t-\tau_1))+cg(dx_2(t-\tau_2)),\\
\dot{x}_2(t) &= -x_2(t)-ag(bx_2(t-\tau_1))+cg(dx_1(t-\tau_2).,
\end{cases}
\tag{23}
$$

The variables $x_1(t)$ and $x_2(t)$ represent the population-averaged neural activity at time t in layers one and two, respectively. The parameter $a>0$ is a measure of the strength of inhibitory feedback, while $c>0$ measures the strength of the excitatory effect of one layer on the other. The parameters $b>0$ and $d>0$ are saturation rates and the delays $\tau_{1,2}$ represent time lags in the inhibitory feedback loop and excitatory inter-layer connection. Note that the system is symmetric with respect to interchanging the labels 1 and 2, so equilibria are necessarily of the form (x_0,x_0). The function g is smooth, strictly increasing and satisfies $g(0)=0$ and $g'(0)=1$. In accordance with [Janssens, 2007] we fix the numerical values

$$
b=2.0,\qquad d=1.2,\qquad \tau_1=12.7,\qquad \tau_2=20.2
$$

and take for $g:\mathbb{R}\to\mathbb{R}$ the sigmoidal form

$$
g(z)=[\tanh(z-1)+\tanh(1)]\cosh(1)^2.
$$

For more information about the system and the meaning of the parameters we refer to [Visser et al., 2012] and [Visser, 2013]. We note that in [Visser et al., 2012] the time delays are set differently to $\tau_1=11.6$ and $\tau_2=20.3$.

The files of this demonstration are located in the folder `demos/nmfm_demo` of the DDE-BifTool package.

7.1 Initialization

First, we load the DDE-BifTool core, the DDE-BifTool utilities and the normal form extension. Then we initialize the `funcs` structure.

```
clear variables;
close all;
addpath('../../ddebiftool',...
    '../../ddebiftool_utilities',...
    '../../ddebiftool_extra_nmfm');
funcs=set_funcs(...
    'sys_rhs', @(xx,par) sys_rhs(xx,par),...
    'sys_tau', @() sys_tau(),...
    'sys_deri', @(xx,par,nx,np,v) sys_deri(xx,par,nx,np,v),...
    'sys_mfderi', @(xx,par,varargin) sys_mfderi(xx,par,varargin{:}));
```

The function `sys_mfderi` computes the multilinear forms of the system needed for the calculation of the normal form coefficients, see (7). In case `sys_mfderi` is not provided the higher order derivatives will be approximated calculated with finite difference. The Mathematica notebook `genr_sys.mth` available with DDE-BifTool, only generate the first order derivatives. To generate the function `sys_mfderi` a new Maple script `Maple_gen_sys.mw` has been developed by Bram Wage. For usage of the script we refer to the script itself.

7.2 Continuation of steady state point

We set up a steady state branch using the function `SetupStst` from the utilities and do a standard continuation.

```
% Define the boundaries and step sizes
% of the active parameters.
astepsize = 0.005;
cstepsize = 0.05;
amin = 0.0;
amax = 0.55;
cmin = 0.0;
cmax = 1.0;

% continue the trivial steady-state
a=0.25; b=2.0; c=15/29; d=1.2; tau1=12.7; tau2=20.2;
par = [a, b, c, d, tau1, tau2];
branch1 = SetupStst(funcs,'x',[0;0],'parameter',par,...
        'contpar',1,'max_step',[1 astepsize],...
        'max_bound',[1 amax],'min_bound',[1 amin],...
        'newheuristics_tests',0);
branch1.method.continuation.plot = 0;
[branch1,s,f,r] = br_contn(funcs,branch1,300);
branch1 = br_rvers(branch1);
[branch1,s,f,r] = br_contn(funcs,branch1,300);

% plot branch1
figure; [xm,ym] = df_measr(0,branch1,1);
br_plot(branch1,xm,ym);
xlabel('a'); ylabel('x_1');
```

The plot of `branch1` is shown in Figure 1.

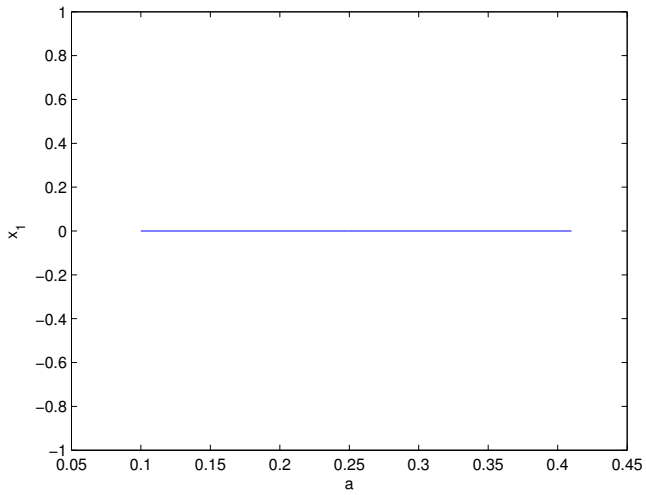

Figure 1: Plot of `branch1`

7.3 Bifurcation detection on the steady state branch

Before calling the bifurcation detection script `br_bifdet` we first have to calculate the stability information

```
branch1.method.bifurcation.plot_testfunctions = 1;
branch1.method.bifurcation.monitor_eigenvalues = 0;
branch1.method.bifurcation.pause_on_bifurcation= 0;
branch1.method.bifurcation.imagthreshold = 0;
branch1.method.stability.maximal_time_step=0.01;
branch1.method.stability.minimal_real_part=-0.01;

fprintf('Calculating stability\n');
branch1 = br_stabl(funcs,branch1,0,0);
fprintf('Bifurcation detection\n');
branch1 = br_bifdet(funcs, branch1);
```

in the `Matlab` console we see the output

```
Calculating stability
Bifurcation detection
BR_BIFDET: Hopf detected near par(1) = 0.3000000000.
BR_BIFDET: Hopf located at  par(1) = 0.2988605971.
BR_BIFDET: Normal form coefficient: L1 = -0.0054576675

BR_BIFDET: Hopf detected near par(1) = 0.3250000000.
BR_BIFDET: the detected hopf point does not fall within the branch.

BR_BIFDET: Hopf detected near par(1) = 0.3500000000.
BR_BIFDET: Hopf located at  par(1) = 0.3453112603.
BR_BIFDET: Normal form coefficient: L1 = -0.0133009607

BR_BIFDET: Hopf detected near par(1) = 0.3600000000.
BR_BIFDET: the detected hopf point does not fall within the branch.

BR_BIFDET: Hopf detected near par(1) = 0.3700000000.
BR_BIFDET: Hopf located at  par(1) = 0.3676281680.
BR_BIFDET: Normal form coefficient: L1 = -0.0028635604
```

```
BR_BIFDET: Hopf detected near par(1) = 0.4300000000.
BR_BIFDET: the detected hopf point does not fall within the branch.

BR_BIFDET: Hopf detected near par(1) = 0.4950000000.
BR_BIFDET: Hopf located at  par(1) = 0.4923372193.
BR_BIFDET: Normal form coefficient: L1 = -0.0024530632

BR_BIFDET: Hopf detected near par(1) = 0.5350000000.
BR_BIFDET: the detected hopf point does not fall within the branch.
```

The Hopf points detected that don't fall within the branch are explained by looking at the test function ψ_H plotted in Figure 2. We see discontinuities when the smallest real part of the complex eigenvalues being followed, changes. This yields in a sign change when there is no Hopf point.

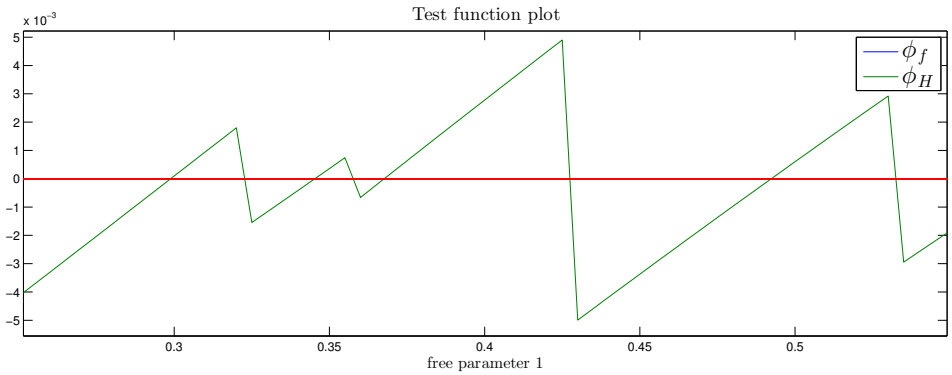

Figure 2: Plot of test functions for `branch1`

7.4 Flagged points on `branch1`

Four Hopf points are detected and located. These points are flagged in the branch, e.g. `branch1.point(10).flag = 'hopf'`. Furthermore normal form information has been added for these points: `branch1.point(10).nmfm.L1 = -0.0055`. We will choose two Hopf points and continue them in the parameters a and c. To this end, we need to know where the bifurcations were found. We do this by calling `br_getflags` on the steady state branch

```
>> FPI = br_getflags(branch1)
10    20    24    49
```

7.5 Continuation of the first Hopf point

We do a standard continuation, using the first starting index obtained from the flagged point indices

```
astepsize = 0.05; % decrease stepsize a
FPI = br_getflags(branch1);
start_ind = FPI(bif2num('hopf'),1);

[hopf_branch1, suc] = SetupHopf(funcs, branch1, start_ind,...
    'contpar',[1 3], 'dir', 3, 'step', cstepsize);
```

```
hopf_branch1.parameter.min_bound=[5 0; 6 0; 3 cmin; 1 amin];
hopf_branch1.parameter.max_bound=[1 amax; 3 cmax];
hopf_branch1.parameter.max_step=[1 0.002; 3 cstepsize];
hopf_branch1.method.continuation.plot = 0;
hopf_branch1.method.stability.minimal_time_step = 0.005;
[hopf_branch1,s,f,r]=br_contn(funcs,hopf_branch1,300);
hopf_branch1 = br_rvers(hopf_branch1);
[hopf_branch1,s,f,r]=br_contn(funcs,hopf_branch1,300);
```

7.6 Detection of bifurcations on the Hopf branch

We repeat the steps for bifurcation detection, but now apply it to the Hopf branch. This yields several higher-order bifurcations

```
>> hopf_branch1.method.stability.minimal_real_part = -0.03;
>> hopf_branch1 = br_stabl(funcs,hopf_branch1,0,0);
>> hopf_branch1 = br_bifdet(funcs,hopf_branch1);
BR_BIFDET: Zero Hopf detected near par(1) = 0.0141129425, par(3) = 0.8546265100.
BR_BIFDET: omega = 0.2958047065, par(1) = 0.0132898339, par(3) = 0.8554830515.
BR_BIFDET: s = 0.0000803554, theta = 2.0966153011.

BR_BIFDET: Generalized Hopf detected near par(1) = 0.0141129425, par(3) = 0.8546265100.
BR_BIFDET: Failed to correct generalized Hopf.

BR_BIFDET: Double-Hopf detected near par(1) = 0.0921713247, par(3) = 0.7712365372.
BR_BIFDET: omega1 = 0.2899790050, omega2 = 0.1560400855, par(1) = 0.0903758907, par(3) = 0.7732081015.
BR_BIFDET: theta = 1.6991967765, delta = 1.5262842690.
BR_BIFDET: The eigenvalues are
        0.0000 + 0.1560i
        0.0000 - 0.1560i
        0.0000 + 0.2900i
        0.0000 - 0.2900i

BR_BIFDET: Generalized Hopf detected near par(1) = 0.2523811833, par(3) = 0.5805259207.
BR_BIFDET: L2 = 0.0011032785, omega = 0.2759094349, par(1) = 0.2518651159, par(3) = 0.5812019758.

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  2.775558e-17.
> In nmfm_border at 26
  In nmfm_hopf at 75
  In br_bifdet at 202

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  2.775558e-17.
> In nmfm_border at 28
  In nmfm_hopf at 75
  In br_bifdet at 202
```

Since the characteristic matrix in (12) becomes close to the zero matrix near the parameter values $(a, c) \approx (0.5130, 0)$, the calculations for the normal form coefficients of a Hopf point produce warning messages.

The falsely detected generalized Hopf point at the zero-Hopf point with parameter values $(a, c) = (0.0141129425, 0.8546265100)$ can be explained by comparing the test function $\psi_{FH}^{H} = \det(\Delta(0))$, which vanishes at a zero-Hopf point, and the calculation of the term h_{11} in (12). Without going into detail we see in Figure 3 that the test function for the generalized Hopf point goes to $\pm\infty$ at the zero-Hopf point.

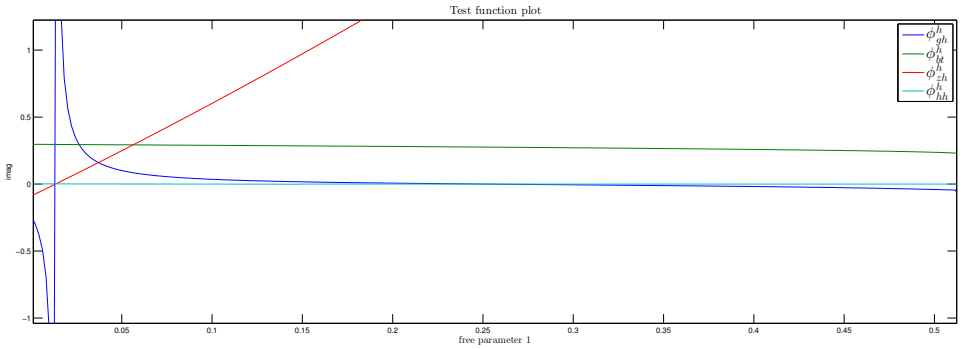

Figure 3: Plot of the test functions for `hopf_branch1`

7.7 The second Hopf branch

We repeat the same procedure for second Hopf branch

```
start_ind = FPI(bif2num('hopf'),2);
[hopf_branch2, suc] = SetupHopf(funcs, branch1, start_ind,...
    'contpar', [1 3], 'dir', 3, 'step', cstepsize);
hopf_branch2.parameter.min_bound=[5 0; 6 0; 3 cmin; 1 amin];
hopf_branch2.parameter.max_bound=[1 amax; 3 cmax];
hopf_branch2.parameter.max_step=[1 0.002; 3 cstepsize];
hopf_branch2.method.continuation.plot = 0;
hopf_branch2.method.stability.minimal_time_step = 0.005;
[hopf_branch2,s,f,r]=br_contn(funcs,hopf_branch2,300);
hopf_branch2 = br_rvers(hopf_branch2);
[hopf_branch2,s,f,r]=br_contn(funcs,hopf_branch2,300);
hopf_branch2.method.stability.minimal_real_part = -0.03;
hopf_branch2 = br_stabl(funcs,hopf_branch2,0,0);
hopf_branch2 = br_bifdet(funcs, hopf_branch2);
```

which generates the output

```
BR_BIFDET: Zero Hopf detected near par(1) = 0.0045472202, par(3) = 0.8391205690.
BR_BIFDET: omega = 0.1485574983, par(1) = 0.0038001792, par(3) = 0.8396669570.
BR_BIFDET: s = 0.0000855576, theta = 2.0243171041.

BR_BIFDET: Generalized Hopf detected near par(1) = 0.0045472202, par(3) = 0.8391205690.
BR_BIFDET: Failed to correct generalized Hopf.

BR_BIFDET: Double-Hopf detected near par(1) = 0.0906215708, par(3) = 0.7730096131.
BR_BIFDET: omega1 = 0.1560400907, omega2 = 0.2899790005, par(1) = 0.0903759486, par(3) = 0.7732080509.
BR_BIFDET: The eigenvalues are
   0.0000 + 0.2900i
   0.0000 - 0.2900i
  -0.0000 + 0.1560i
  -0.0000 - 0.1560i

BR_BIFDET: Generalized Hopf detected near par(1) = 0.2568681881, par(3) = 0.6208054350.
BR_BIFDET: L2 = 0.0018108424, omega = 0.1722172445, par(1) = 0.2566152979, par(3) = 0.6210718434.

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  2.775558e-17.
> In nmfm_border at 26
  In nmfm_hopf at 75
  In br_bifdet at 182
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  2.775558e-17.
```

```
> In nmfm_border at 28
  In nmfm_hopf at 75
  In br_bifdet at 182

BR_BIFDET: Double-Hopf detected near par(1) = 0.5130096560, par(3) = 0.0000000000.
BR_BIFDET: omega1 = 0.2295988426, omega2 = 0.2295988426, par(1) = 0.5130096560, par(3) = 0.0000000002.
BR_BIFDET: theta = 2.0000000000, delta = 2.0000000000.
BR_BIFDET: The eigenvalues are
        0.0000 + 0.2296i
        0.0000 - 0.2296i
        0.0000 + 0.2296i
        0.0000 - 0.2296i
```

As in the previous subsection there is a falsely detected generalized Hopf at a zero-Hopf point. Also the warning messages appear again as expected near $(a, c) \approx (0.5130, 0)$. There is a double-Hopf located at $(a, c) \approx (0.5130, 0)$. The eigenvalues show there is 1:1 resonance. Therefore the computed normal form coefficients do not apply here.

7.8 The Fold branch through the zero-Hopf

To demonstrate the detection of zero-Hopf points on fold curves we continue the fold curve emanating from the zero-Hopf point detected on `hopf_branch1`.

```
%% Continue Fold emanating from Zero Hopf
FPI2 = br_getflags(hopf_branch1);
start_ind = FPI2(bif2num('ZH'));
[fold_branch, suc] = SetupFold(funcs, hopf_branch1, start_ind, ...
    'contpar', [1 3], 'dir', 3, 'step', cstepsize);

cmin=0; cmax=1; amin=0; amax=1;

fold_branch.parameter.min_bound=[3 cmin; 1 amin];
fold_branch.parameter.max_bound=[1 amax; 3 cmax];
fold_branch.parameter.max_step=[1 0.001; 3 0.001];

[fold_branch,s,f,r]=br_contn(funcs,fold_branch,300);
fold_branch = br_rvers(fold_branch);
[fold_branch,s,f,r]=br_contn(funcs,fold_branch,300);

fold_branch = br_stabl(funcs,fold_branch,0,0);

fold_branch.method.bifurcation.monitor_eigenvalues = 1;
fold_branch.method.bifurcation.pause_on_bifurcation= 0;
fold_branch = br_bifdet(funcs,
fold_branch); FPI=br_getflags(fold_branch);
```

Which generates the output:

```
BR_CONTN warning: boundary hit.
BR_CONTN warning: boundary hit.
BR_BIFDET: Zero-Hopf detected near par(1) = 0.0684898635, par(3) = 0.9474830977.
BR_BIFDET: Zero Hopf located at par(1) = 0.0689934962, par(3) = 0.9483224999.
BR_BIFDET: s = 0.0000554219, theta = 2.5704064406.

BR_BIFDET: Zero-Hopf detected near par(1) = 0.0672898543, par(3) = 0.9454830980.
BR_BIFDET: Failed to correct zero-Hopf.

BR_BIFDET: Zero-Hopf detected near par(1) = 0.0660898775, par(3) = 0.9434831017.
```

```
BR_BIFDET: Zero Hopf located at par(1) = 0.0661546818, par(3) = 0.9435911421.
BR_BIFDET: s = 0.0000583652, theta = 2.3917304978.

BR_BIFDET: Zero-Hopf detected near par(1) = 0.0564898556, par(3) = 0.9274830981.
BR_BIFDET: Failed to correct zero-Hopf.

BR_BIFDET: Zero-Hopf detected near par(1) = 0.0132898339, par(3) = 0.8554830964.
BR_BIFDET: Zero Hopf located at par(1) = 0.0132898452, par(3) = 0.8554830917.
BR_BIFDET: s = 0.0000803554, theta = 2.0966153774.

BR_BIFDET: Zero-Hopf detected near par(1) = 0.0084898655, par(3) = 0.8474830901.
BR_BIFDET: Failed to correct zero-Hopf.

BR_BIFDET: Zero-Hopf detected near par(1) = 0.0036898645, par(3) = 0.8394830954.
BR_BIFDET: Zero Hopf located at par(1) = 0.0038001703, par(3) = 0.8396669952.
BR_BIFDET: s = 0.0000855576, theta = 2.0243170475.
```

The false detection of the zero-Hopf points is caused by the chaining of the eigenvalues with the smallest real part in the function `nmfm_smrp`. This can be seen by monitoring the eigenvalues. The last two located zero-Hopf points are the same as the ones located on the Hopf branches. In addition two extra zero-Hopf points are detected. The Hopf branches emanating from these points could be continued as well. However no new insights regarding detection, location and normal form computation would be giant. Therefore we will stop our analysis here.

7.9 Plotting the bifurcations on the branches

We are now ready to make a bifurcation diagram. To visualize the different bifurcation points we use the function `br_bifplot`.

```
figure;

bifplot = df_bifplot(); bifplot.zeho = '*';
bifplot.hoho = 'o';
bifplot.genh = '+';

[xm,ym] = df_measr(0,hopf_branch1,1);
br_plot(hopf_branch1,xm,ym,'-');
br_bifplot(hopf_branch1,xm,ym,bifplot);

[xm,ym] = df_measr(0,hopf_branch2,1);
br_plot(hopf_branch2,xm,ym,'-');
br_bifplot(hopf_branch2,xm,ym,bifplot);

[xm,ym] = df_measr(0,fold_branch,1);
br_plot(fold_branch,xm,ym,'-g');
br_bifplot(fold_branch,xm,ym,bifplot);

br_plot(branch1,xm,ym,'--');

xlabel('a'); ylabel('c');
```

See Figure 4 for the resulting bifurcation diagram.

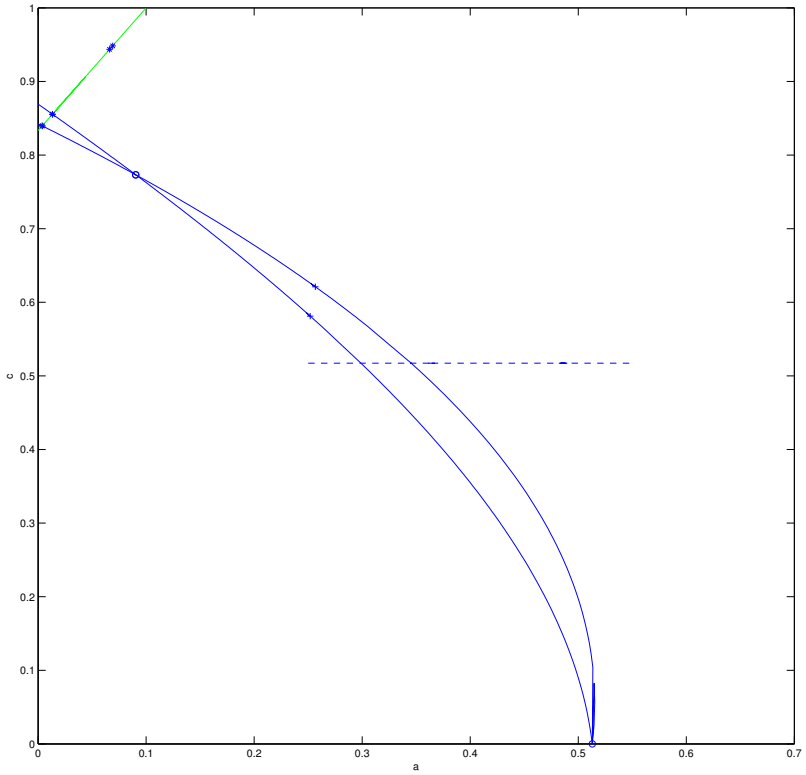

Figure 4: Plot of detected bifurcations on the fold and Hopf branches. The asterisks '*' indicates fold-Hopf bifurcations, the open circle 'o' indicates a double-Hopf, the plus '+' indicates generalized Hopf.

8 Example with Bogdanov-Takens bifurcation

In [Liu et al., 2013] the Bogdanov-Takens bifurcation of the following delayed predator prey system

$$\begin{cases} \dot{x} &= rx\left(1-\dfrac{x}{K}\right)-\dfrac{\alpha y(x-\bar{m})}{Ay+x-\bar{m}}-\bar{h}, \\ \dot{y} &= sy\left(1-\dfrac{dy(t-\bar{\tau})}{x(t-\bar{\tau})-\bar{m}}\right), \end{cases} \tag{24}$$

is considered. Here x and y stand for prey and predator population (or densities) at time t, respectively. The predator growth is of logistic type with growth rate r and carrying capacity K in the absence of predation; α and A stand for the predator capturing rate and half saturation constant, respectively; s is the intrinsic growth rate of predator; however, carrying capacity x/b (b is the conversion rate of prey into predators) is the function on the population size of

prey. The parameters $\alpha, A, \bar{m}, \bar{h}, s, b$, and $\bar{\tau}$ are all positive constants. $\bar{m}$ is a constant number of prey using refuges; h is the rate of prey harvesting. System (24) can be transformed into

$$\begin{cases} \dot{x} &= (x+m)(1-x-m) - \dfrac{xy}{ay+x} - h, \\ \dot{y} &= \delta y \left(\beta - \dfrac{y(t-\tau)}{x(t-\tau)}\right), \end{cases} \tag{25}$$

see [Liu et al., 2013] for the transformation and the meaning of the new parameters.

Let

$$\begin{aligned} 0 < m < \frac{1}{2}\left(1 - \frac{\beta}{a\beta+1}\right), \\ h = \frac{1}{4}\left(\frac{\beta}{a\beta+1} - 1\right)^2 + \frac{m\beta}{a\beta+1}. \end{aligned} \tag{26}$$

Then $P_* = (x_*, y_*)$ is an interior positive equilibrium point of systems (25), where

$$x_* = -\frac{1}{2}\left(\frac{\beta}{a\beta+1} + 2m - 1\right), \qquad y_* = \beta x_*. \tag{27}$$

In order to discuss the properties of system (25) in the neighborhood of the equilibrium point $P_* = (x_*, y_*)$, let $x = x - x_*$, $y = y - y_*$; then P_* is translated to $(0,0)$, and the linearization of system (25) becomes (still denoting $\bar{x}$, $\bar{y}$ as x, y)

$$\begin{cases} \dot{x} = \dfrac{\tau\beta}{(a\beta+1)^2}x - \dfrac{\tau}{(a\beta+1)^2}y, \\ \dot{y} = \tau\delta\beta^2 x(t-1) - \tau\delta\beta y(t-1). \end{cases} \tag{28}$$

Here we have rescaled the time as $t \to \dfrac{t}{\tau}$, so that τ can be treated as an ordinary parameter. Denoting $\mathbf{x} \equiv (x, y)$ system (28) can be written as

$$\dot{\mathbf{x}} = \frac{\tau}{(a\beta+1)^2}\begin{pmatrix} \beta & -1 \\ 0 & 0 \end{pmatrix}\mathbf{x} + \tau\delta\beta\begin{pmatrix} 0 & 0 \\ \beta & -1 \end{pmatrix}\mathbf{x}(t-1).$$

The characteristic matrix becomes

$$\Delta(\lambda) = \begin{pmatrix} \lambda - \dfrac{\tau\beta}{(a\beta+1)^2} & \dfrac{\tau}{(a\beta+1)^2} \\ -\tau\delta\beta^2 e^{-\lambda} & \lambda + \tau\delta\beta e^{-\lambda} \end{pmatrix}.$$

It follows that the characteristic equation of (28) is given by

$$\det(\Delta(\lambda)) = \lambda^2 + \beta\tau\left(\delta e^{-\lambda\tau} - \frac{1}{(a\beta+1)^2}\right)\lambda. \tag{29}$$

Expanding (29) around $\lambda = 0$ yields

$$\det(\Delta(\lambda)) = \beta\tau\left(\delta - \frac{1}{(a\beta+1)^2}\right)\lambda + 2\left(1 - \beta\delta\tau\right)\lambda^2 + \mathcal{O}(\lambda^3).$$

Clearly, if

$$\begin{aligned} \delta &= \frac{1}{(a\beta+1)^2}, \\ \tau \neq \frac{1}{\delta\beta}, \qquad &\text{that is,} \qquad \tau \neq \frac{(a\beta+1)^2}{\beta}; \end{aligned} \tag{30}$$

then $\lambda = 0$ is a double zero eigenvalue.

We now assume that we are at a Bogdanov-Takens point such that (30) are satisfied. Then the vectors

$$\begin{aligned} q_0 &= \left(\frac{1}{\sqrt{\beta^2+1}}, \frac{\beta}{\sqrt{\beta^2+1}} \right)^T, \\ q_1 &= \left(\frac{a^2\beta - \frac{\beta\tau^2}{2(a\beta+1)^2-2\beta\tau} + 2a + \frac{1}{\beta} - \tau}{\sqrt{\beta^2+1}}, \frac{\beta\tau\left(\beta\tau - 2(a\beta+1)^2\right)}{2\sqrt{\beta^2+1}\left((a\beta+1)^2-\beta\tau\right)} \right)^T, \\ p1 &= \left(\frac{\beta\sqrt{\beta^2+1}}{(a\beta+1)^2-\beta\tau}, -\frac{\sqrt{\beta^2+1}}{(a\beta+1)^2-\beta\tau} \right)^T, \\ p_0 &= \left(0, \frac{\sqrt{\beta^2+1}}{\beta} \right)^T, \end{aligned}$$

satisfy the relations (18) and (19). Using a computer algebra system like `Mathematica`, the coefficients in (20) become

$$\begin{aligned} a &= -\frac{\beta}{\sqrt{\beta^2+1}\left((a\beta+1)^2-\beta\tau\right)}, \\ b &= \frac{4\beta\tau(a\beta+1)^2 - 2(a\beta+1)^4 - \beta^2\tau^2}{\sqrt{\beta^2+1}\left((a\beta+1)^2-\beta\tau\right)^2}. \end{aligned} \tag{31}$$

We note that although these coefficients are slightly different from the coefficients derived in [Liu et al., 2013], the sign of the product of ab remains the same. Here we choose to normalize the vector q_0 to have length 1, as done in the Matlab scripts, see the file `p_tobt.m`. Furthermore, the coefficients here have been derived from system (25) directly, whereas in [Liu et al., 2013] first the time as been rescaled by $t \to \frac{t}{\tau}$.

The files of this demonstration are located in the folder `demos/Holling-Tanner` of the DDE-BifTool package.

8.1 Generating system files for DDE-BifTool

To generate the system files we again use the `Maple` script `Maple_gen_sys.mw`. We only display the file `sys_rhs.m` here, since the files `sys_deri` and `sys_mfderi` are too long

```
function f = sys_rhs(xx,par)

f(1,1) = (xx(1,1)+par(4))*(1-xx(1,1)-par(4)) -...
```

```
    xx(1,1)*xx(2,1)/(par(3)*xx(2,1)+xx(1,1))-par(5);
f(2,1) = par(6)*xx(2,1)*(par(1)-xx(2,2)/xx(1,2));

end
```

8.2 Initialization of the system in DDE-BifTool

We add the necessary paths to our workspace so that we can access the script files of `ddebiftool`, `ddebiftool_utilities`, `ddebiftool_extra_nmfm`. We also add the extension `ddebiftool_extra_psol` for the continuation of the homoclinic orbit emanating from the Bogdanov-Takens point .

```
clear;
close all;
addpath('../../../ddebiftool',...
    '../../../ddebiftool_extra_psol',...
    '../../../ddebiftool_extra_nmfm',...
    '../../../ddebiftool_utilities');
funcs=set_funcs(...
    'sys_rhs', @(xx,par) sys_rhs(xx,par),...
    'sys_tau', @() 2,...
    'sys_deri', @(xx,par,nx,np,v) sys_deri(xx,par,nx,np,v),...
    'sys_mfderi',@(xx,par,varargin) sys_mfderi(xx,par,varargin{:}));

disp('Holling-Tanner');
% par=(beta,tau,a,m,h,delta)
```

8.3 Continuation of steady-state branch

We fixed the parameters

$$\begin{array}{lll} a = 0.5, & \tau = 0.78125, & m = 0.02, \\ h = 0.098, & \beta = 0.4, & \delta = 0.54089. \end{array} \tag{32}$$

Then we continue the positive steady-state (27) in β.

```
beta=0.5;a=0.5;
m=(1/30)*(1-beta/(a*beta+1));
h=(1/4)*(beta/(a*beta+1)-1)^2+m*beta/(a*beta+1);
tau=1/4*(a*beta+1)^2/beta; delta=1/(a*beta+1)^2;

beta=0.4;
delta=0.5409;

stst.parameter=[beta,tau,a,m,h,delta];

xster=-(1/2)*((beta/(a*beta+1)+2*m-1)+...
        sqrt((1-2*m-beta/(a*beta+1))^2+4*(m*(1-m)-h)));
yster=beta*xster;

contpar=1;
stst_branch = SetupStst(funcs,'x',[xster; yster],'parameter',stst.parameter,...
    'contpar',contpar,'max_step',[contpar 0.02],'min_bound',...
```

```
    [contpar 0.4],'max_bound',[contpar 0.6],...
    'newheuristics_tests',0);

stst_branch.method.continuation.plot = 1;

[stst_branch,s,f,r] = br_contn(funcs,stst_branch,100);

stst_branch.method.bifurcation.monitor_eigenvalues=0;
stst_branch.method.bifurcation.plot_testfunctions=1;
stst_branch.method.bifurcation.pause_on_bifurcation=0;

stst_branch = br_stabl(funcs,stst_branch,0,0);
stst_branch=br_bifdet(funcs,stst_branch);
```

Running the above code outputs:

```
BR_CONTN warning: boundary hit.

BR_BIFDET: Fold detected near par(1) = 0.4998182859.
BR_BIFDET: Fold located at  par(1) = 0.5000000000.
BR_BIFDET: Normal form coefficient: b = 4.8721895296

BR_BIFDET: Hopf detected near par(1) = 0.4916389145.
BR_BIFDET: Hopf located at  par(1) = 0.4939507149.
BR_BIFDET: Normal form coefficient: L1 = 192.7259505902
```

In Figure 5 we have plotted the continuation of the steady-state branch and the test functions ψ_f and ψ_H.

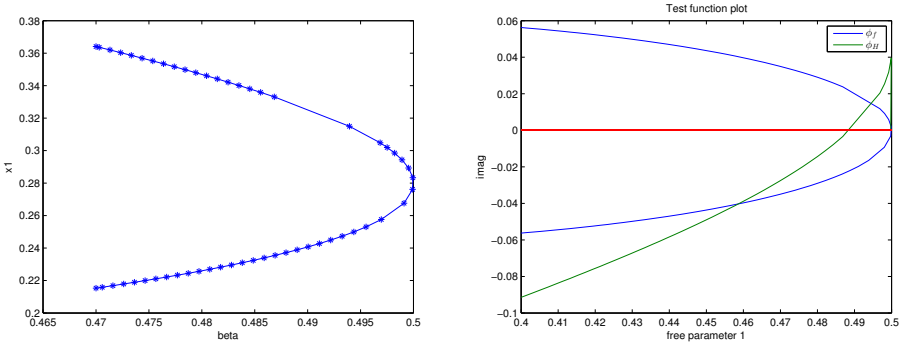

Figure 5: Left is a plot of the continuation of the steady-state point. In the right figure the test functions are plotted. Here we see that the Hopf bifurcation is near the fold bifurcation, as predicted by a Bogdanov-Takens bifurcation.

8.4 Continuation of the Hopf point

We continue the detected Hopf point at $\beta = 0.4886954168$ in (β, δ)

```
FPI = br_getflags(stst_branch);
start_ind = FPI(1);

% We do a standard continuation, using the starting index obtained from the
% flagged point indices.
```

```
fprintf('------ Hopf branch ------\n');
[hopf_branch, suc] = SetupHopf(funcs, stst_branch, start_ind,...
    'contpar', [1 6], 'dir', 1, 'step', 0.002);

betamin=0.4; betamax=0.6; deltamin=0.4; deltamax=0.7;

hopf_branch.parameter.min_bound=[1 betamin; 6 deltamin];
hopf_branch.parameter.max_bound=[1 betamax; 6 deltamax];
hopf_branch.parameter.max_step=[1 0.0005; 6 0.0005];

hopf_branch.method.stability.minimal_time_step = 0.005; % default 0.01
[hopf_branch,s,f,r]=br_contn(funcs,hopf_branch,250);
```

In Figure 6 we see two plots of the Hopf branch. In the first plot we have plotted the two continuation parameters β against δ. In the second plot we have plotted the point number along the branch versus δ. From the two graphs we conclude that the Hopf curve makes a turn at $(\beta, \delta) \approx (0.5, 0.64)$.

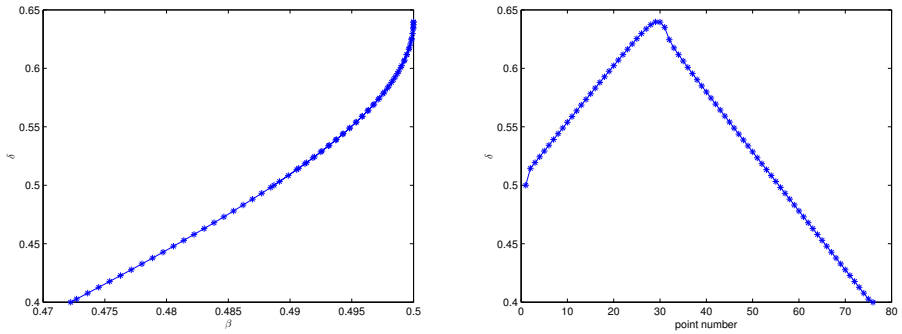

Figure 6: Left is a plot of the continuation of the steady-state point. At the right side we see a plot of the point number along the branch versus δ.

In Figure 7 we have plotted the test functions. We see that the test function for a Bogdanov-Takens point on a Hopf curve passes 0 even when the parameters make a turn.

8.5 Detection on the Hopf branch

We use the function `br_bifdet` to detect bifurcations on the Hopf branch

```
>> hopf_branch = br_stabl(funcs,hopf_branch,0,0);
>> [hopf_branch,success] = br_bifdet(funcs,hopf_branch);
BR_BIFDET: Bogdanov-Takens detected near par(1) = 0.4999998471, par(6) = 0.6395308481.
BR_BIFDET: a = -0.3812625439, b = -1.6892715659, par(1) = 0.5000000000, par(6) = 0.6400000000.
```

As predicted we see a Bogdanov-Takens point (BT point) located at $(\beta, \delta) = (0.5, 0.64)$. From the normal form coefficients we conclude that the Hopf bifurcation generates an unstable limit cycle. Combining $(\beta, \delta) = (0.5, 0.64)$ with (32) we confirm that the equations for a Bogdanov-Takens point derived analytically in (30) and the coefficients in (31) are satisfied.

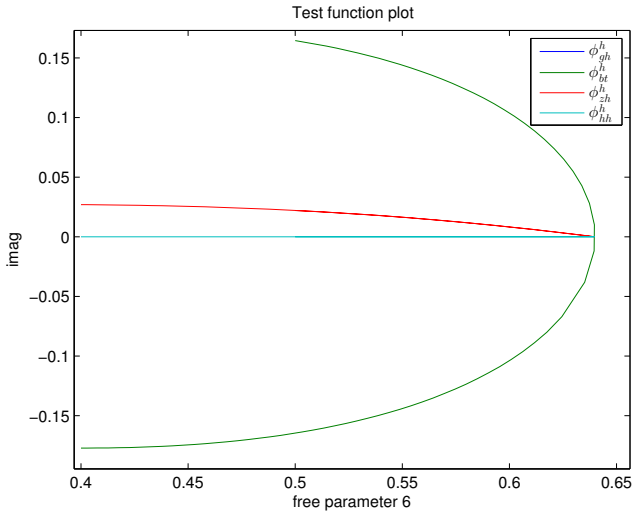

Figure 7: Plot of the test function along`hopf_branch`.

8.6 Direct calculation of the Bogdanov-Takens normal form coefficients

Since in this example we know the position of the Bogdanov-Takens through analytic calculations, we could calculate the normal form coefficients directly as shown below.

```
beta=0.5;a=0.5;
m=(1/30)*(1-beta/(a*beta+1));
h=(1/4)*(beta/(a*beta+1)-1)^2+m*beta/(a*beta+1);
tau=1/4*(a*beta+1)^2/beta;
delta=1/(a*beta+1)^2;

xster=-(1/2)*(beta/(a*beta+1)+2*m-1);
yster=beta*xster;

p.kind='stst';
p.parameter=[beta,tau,a,m,h,delta];
p.x=[xster;yster];

bt=p_tobt(funcs,p);
bt=nmfm_bt(funcs,bt);
bt.nmfm
```

The last line gives the following output:

```
ans =
        a2: -0.3816
        b2: -1.6895
```

Here $a = a_2$ and $b = b_2$.

8.7 Continue fold point

We will continue the fold point detected on the steady-state.

```
FPI = br_getflags(stst_branch);
start_ind = FPI(bif2num('fold'),1);

[fold_branch, suc] = SetupFold(funcs, stst_branch, start_ind, ...
    'contpar', [1 6], 'dir', 6, 'step', 0.005);
betamin=0.4; betamax=0.6; deltamin=0.3; deltamax=0.7;

fold_branch.parameter.min_bound=[1 betamin; 6 deltamin];
fold_branch.parameter.max_bound=[1 betamax; 6 deltamax];
fold_branch.parameter.max_step=[1 0.005; 6 0.005];

[fold_branch,s,f,r]=br_contn(funcs,fold_branch,300);
fold_branch = br_rvers(fold_branch);
[fold_branch,s,f,r]=br_contn(funcs,fold_branch,300);

fold_branch = br_stabl(funcs,fold_branch,0,0);
[fold_branch,success] = br_bifdet(funcs,fold_branch);
```

In the Matlab console the following output is given:

```
BR_CONTN warning: boundary hit.
BR_CONTN warning: boundary hit.

BR_BIFDET: Cusp detected near par(1) = 0.5000000000, par(6) = 0.6387723214. BR_BIFDET:
Failed to correct cusp.

BR_BIFDET: Bogdanov-Takens detected near par(1) = 0.5000000000, par(6) = 0.6387723214.
BR_BIFDET: a = -0.3812625439, b = -1.6892715659, par(1) = 0.5000000000, par(6) = 0.6400000000.
```

Why there is a falsely detected cusp at near the Bogdanov-Takens point is explained in the next Section, where we treat the cusp bifurcation in detail.

8.8 Homoclinic orbit

We continue the periodic orbit emanating from the Hopf curve in δ

```
hopf=hopf_branch.point(20);
[psol,stp]=p_topsol(funcs,hopf,1e-2,3,27);

ind_delta=6;
mpsol=df_mthod(funcs,'psol');
[psol,s]=p_correc(funcs,psol,ind_delta,stp,mpsol.point);
psol_branch=df_brnch(funcs,ind_delta,'psol');
psol_branch.point=psol;
[psol,stp]=p_topsol(funcs,hopf,1.1e-2,3,27);
[psol,s]=p_correc(funcs,psol,ind_delta,stp,mpsol.point);
psol_branch.point(2)=psol;

figure(2);clf;
[xm,ym]=df_measr(0,psol_branch);
ym.field='period';
ym.col=1;
ym.row=1;
psol_branch.method.continuation.plot_measure.x=xm;
psol_branch.method.continuation.plot_measure.y=ym;
[psol_branch,s,r,f]=br_contn(funcs,psol_branch,40);
```

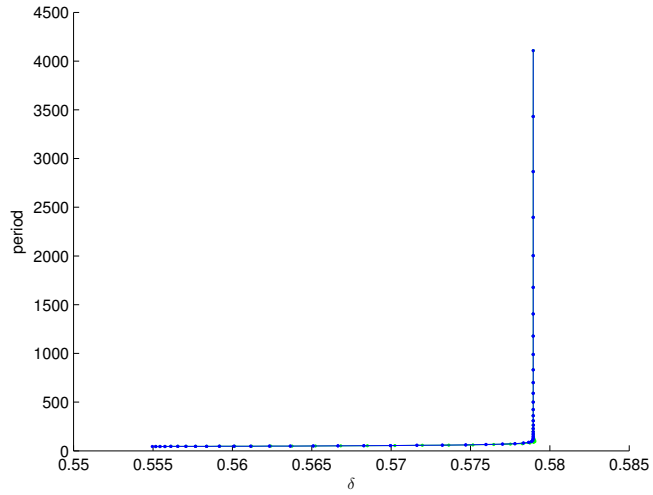

Figure 8: Plot of the cycle emanating from the Hopf curve. At $\delta \approx 0.577$ the period of the cycles goes to infinity, indicating a homoclinic obit.

```
xlabel('$\delta$','interpreter','latex');
ylabel('period');
```

In Figure 8 we have plotted δ against the period of the limit cycle. Around $\delta \approx 0.577$ the period goes to infinity, indicating a homoclinic orbit.

We select the last cycle on the `psol_branch` and convert it to a homoclinic orbit. Then we continue the homoclinic orbit in (β, δ)

```
psol=psol_branch.point(end);
hcli=p_tohcli(funcs,psol);

mhcli=df_mthod(funcs,'hcli');
[hcli,s]=p_correc(funcs,hcli,ind_delta,[],mhcli.point); % correct
hcli=p_remesh(hcli,3,50);    % remesh it and
[hcli,s]=p_correc(funcs,hcli,ind_delta,[],mhcli.point)  % correct it again

%% perturb hcli, correct and continue ind_beta=1;
hcli_br=df_brnch(funcs,[ind_beta, ind_delta],'hcli');
hcli_br.point=hcli;
hcli.parameter(ind_beta)=hcli.parameter(ind_beta)-6e-6;
[hcli,s]=p_correc(funcs,hcli,ind_delta,[],mhcli.point);
hcli_br.point(2)=hcli;

figure(1);
[hcli_br,s,r,f]=br_contn(funcs,hcli_br,28);
hcli_br=br_rvers(hcli_br);
[hcli_br,s,r,f]=br_contn(funcs,hcli_br,40);

xlabel('$\beta$','interpreter','latex');
ylabel('$\delta$','interpreter','latex');
```

In Figure 9 we see the homoclinic orbit continued.

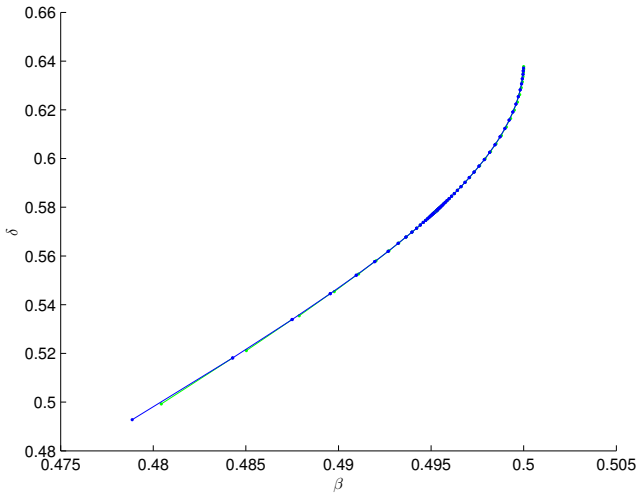

Figure 9: Continuation of the homoclinic orbit in (β, δ).

8.9 Stability of the cycles

The following code calculates the stability of the cycles from `psol_branch`.

```
psol_branch = br_stabl(funcs,psol_branch,0,0);
```

In the field `psol_branch.point(i).stability.mu` the multipliers for the ith point on the branch are given. In Figure 10 we have plotted the multipliers for point number 10. There we see that the cycle is indeed unstable as predicted by the product of the coefficients $ab > 0$.

8.10 Bifurcation diagram

In Figure 11 we have plotted the fold branch as well as the homoclinic and the Hopf branch, giving a complete bifurcation diagram of the Bogdanov-Takens point.

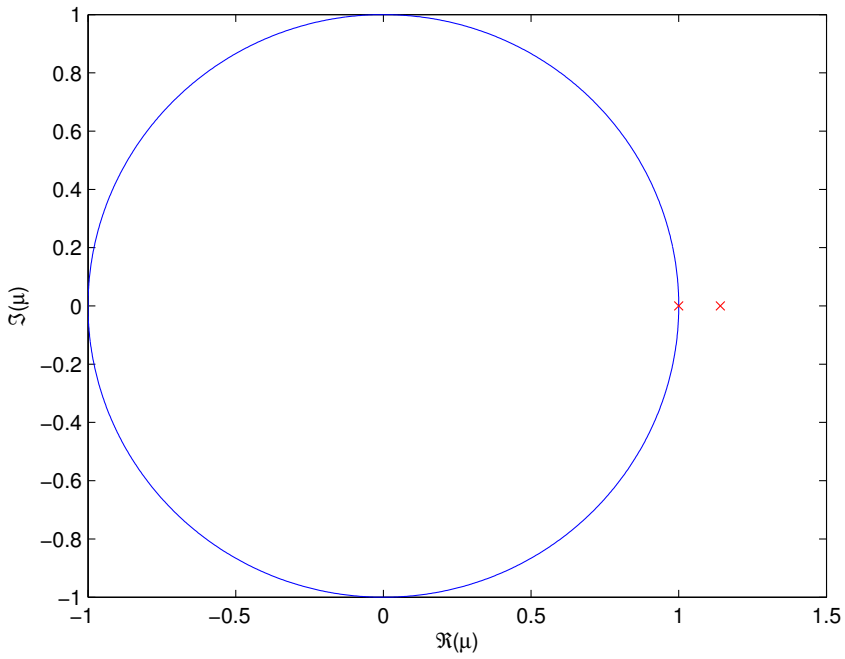

Figure 10: Multipliers for point number 10 from `psol_branch`.

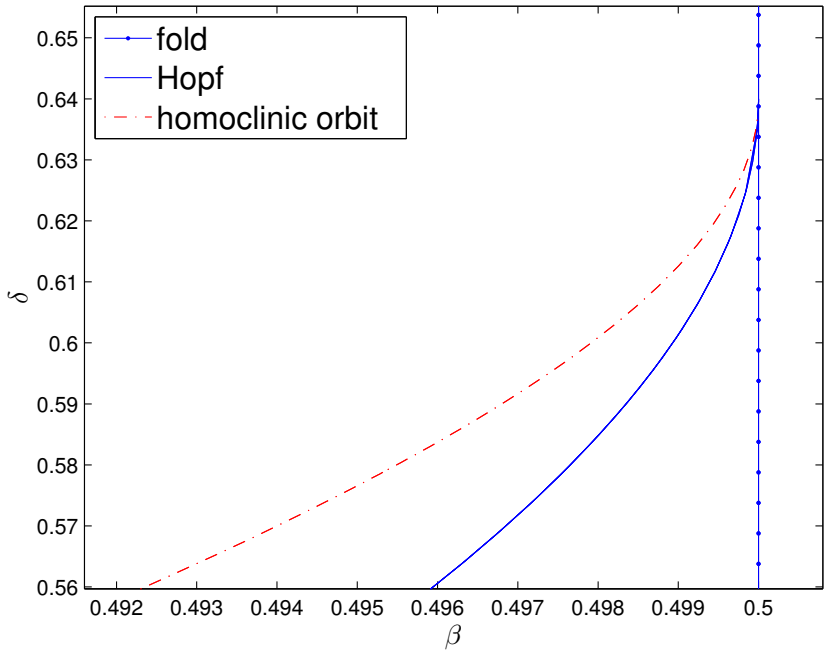

Figure 11: Plot of Hopf, homoclinic and fold branch emanating from the Bogdanov-Takens point.

9 Example with cusp and Bogdanov-Takens bifurcations

In this last example we will consider the mathematical model

$$\begin{cases} \mu \dot{u}_1(t) = -u_1(t) + q_{11}\alpha(u_1(t\text{-}T)) - q_{12}u_2(t\text{-}T) + e_1, \\ \mu \dot{u}_2(t) = -u_2(t) + q_{21}\alpha(u_1(t\text{-}T)) - q_{22}u_2(t\text{-}T) + e_2, \end{cases} \tag{33}$$

which describes the dynamics of a neural network consisting of an excitatory and inhibitory neuron [Giannakopoulos and Zapp, 2001]. The variables and parameters occurring in (33) have the following neurophysiological meaning:

- $u_1, u_2 : \mathbb{R} \to R$ denote the total post-synaptic potential of the excitatory and inhibitory neuron, respectively.
- $\mu > 0$ is a time constant characterizing the dynamical properties of cell membrane.
- $q_{ik} \geq 0$ represents the strength of the connection line from the kth neuron to the ith neuron.
- $\alpha : \mathbb{R} \to \mathbb{R}$ is the transfer function which describes the activity generation of the excitatory neuron as a function of its total potential u_1. The function α is smooth, increasing and has an unique turning point at $u_1 = \theta$. The transfer function corresponding to the inhibitory neuron is assumed to be the identity.
- $T \geq 0$ is a time delay reflecting synaptic delay, axonal and dendritic propagation time.
- e_1 and e_2 are external stimuli acting on the excitatory and inhibitory neuron, respectively.

We consider equation (33) with

$$\alpha(u_1) = \frac{1}{1+e^{-4u_1}} - \frac{1}{2}, \qquad q_{11} = 2.6, \qquad q_{21} = 1.0, \qquad q_{22} = 0.0,$$
$$\mu = 1.0, \qquad T = 1.0, \qquad e_2 = 0.0,$$

and $Q := q_{12}$, $E := e_1$ as bifurcation parameters. Substituting into (33) yields

$$\begin{cases} \dot{u}_1(t) = -u_1(t) + 2.6\alpha(u_1(t\text{-}T)) - Qu_2(t\text{-}T) + E, \\ \dot{u}_2(t) = -u_2(t) + \alpha(u_1(t\text{-}T)). \end{cases} \tag{34}$$

A steady-state solution of (34) is given by $(u_1, u_2) \equiv (0, 0)$ with $E = 0$ and Q arbitrary.

The files of this demonstration are located in the folder `demos/cusp` of the DDE-BifTool package.

9.1 Initialization

As before we start by loading the necessary script files and setting the `funcs` structure.

```
%% setup system clear all;
clear variables close all;
close figures addpath('../../ddebiftool',...
    '../../ddebiftool_extra_psol',...
    '../../ddebiftool_extra_nmfm',...
```

```
    '../../ddebiftool_utilities');

funcs=set_funcs(...
    'sys_rhs', @(xx,par) sys_rhs(xx,par),...
    'sys_tau', @() 6,...
    'sys_deri', @(xx,par,nx,np,v) sys_deri(xx,par,nx,np,v),...
    'sys_mfderi',@(xx,par,varargin) sys_mfderi(xx,par,varargin{:}));
```

9.2 Continuation of steady state point

We continue the steady state $(u_1, u_2) \equiv (0,0)$ at $(Q,E) = (0,0)$ in Q:

```
Q=0; E=0;
q11=2.6; q12=Q; q21=1; e1=E; e2=0; T=1;
par = [q11,q12,q21,e1,e2,T];

stst_br = SetupStst(funcs,'x',[0;0],'parameter',par,...
        'contpar',4,'max_step',[4 0.02],'max_bound',[4 1],'min_bound',[4 0],...
    'newheuristics_tests',0);

[stst_br,s,f,r] = br_contn(funcs,stst_br,300);

stst_br=br_stabl(funcs,stst_br,0,0);
stst_br=br_bifdet(funcs,stst_br);
```

In the matlab console the following output is given:

```
BR_CONTN warning: boundary hit.
BR_BIFDET: Fold detected near par(4) = 0.4899601374.
BR_BIFDET: Fold located at  par(4) = 0.4913668450.
BR_BIFDET: Normal form coefficient: b = 0.7321765321
```

In Figure 12 we have plotted the test functions ψ_f and ψ_H.

The discontinuity in the test function for a Hopf point ϕ_H is produced, since the only complex eigenvalues pass the default lower bound of -1 when calculating the eigenvalues of the points on the steady-state points. This can be seen by monitoring the eigenvalues on the steady-state branch by calling

```
stst_br.method.bifurcation.plot_testfunctions = 1;
stst_br.method.bifurcation.monitor_eigenvalues = 0;
stst_br=br_bifdet(funcs, stst_br);
```

By setting `stst_br.method.stability.minimal_real_part=-5` before calculating the stability along the steady-state branch we get a smooth curve for ψ_H, see Figure (13).

9.3 Continuation of the Fold point and detecting codim-2 bifurcations

We continue the fold point detected on the steady-state branch.

```
FPI=br_getflags(stst_br);
start_ind = FPI(bif2num('fold'),1);

[fold_branch, suc] = SetupFold(funcs, stst_br, start_ind, ...
```

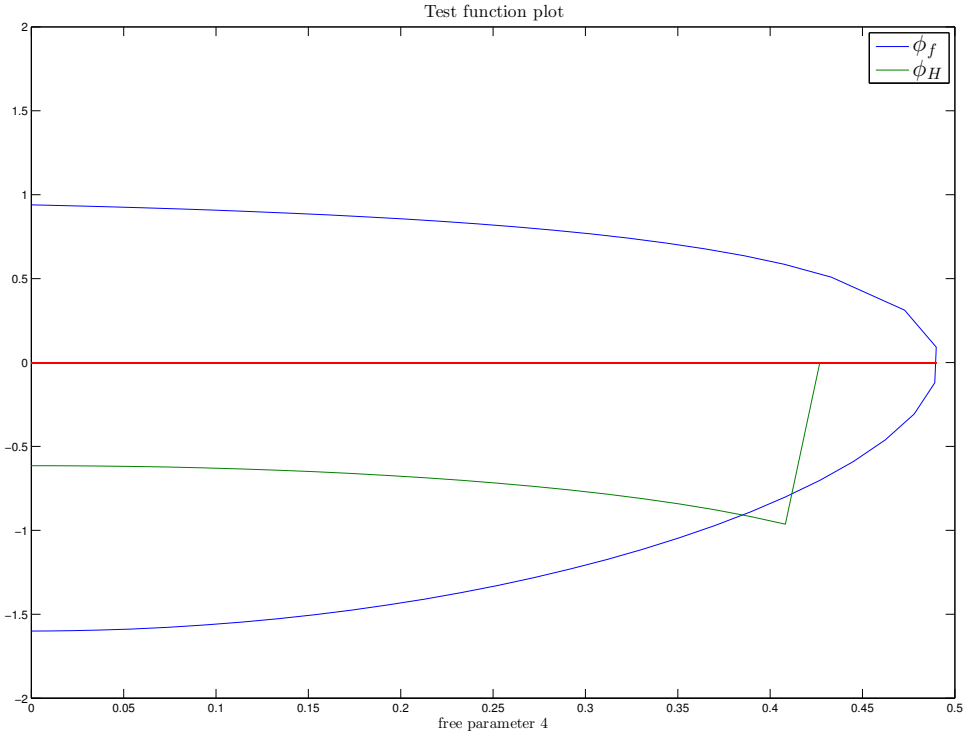

Figure 12: Plot of test function for stst_br

```
        'contpar', [2 4], 'dir', 4, 'step', 0.02);

Qmin=-0.1; Qmax=2; Emin=-0.6; Emax=0.6;
fold_branch.parameter.min_bound=[2 Qmin; 4 Emin];
fold_branch.parameter.max_bound=[2 Qmax; 4 Emax];
fold_branch.parameter.max_step=[2 0.02; 4 0.02];

[fold_branch,s,f,r]=br_contn(funcs,fold_branch,300);
fold_branch = br_rvers(fold_branch);
[fold_branch,s,f,r]=br_contn(funcs,fold_branch,300);

fold_branch = br_stabl(funcs,fold_branch,0,0);
fold_branch=br_bifdet2(funcs,fold_branch);
```

In figure (14) we have plotted the fold branch.

In the matlab console we have the following output:

```
BR_CONTN warning: boundary hit.
BR_CONTN warning: boundary hit.

BR_BIFDET: Bogdanov-Takens detected near par(2) = -0.0908385124, par(4) = 0.5271860637.
Failed to correct Bogdanov-Takens point.

BR_BIFDET: Bogdanov-Takens detected near par(2) = 0.0000000000, par(4) = 0.4913668450.
Failed to correct Bogdanov-Takens point.
```

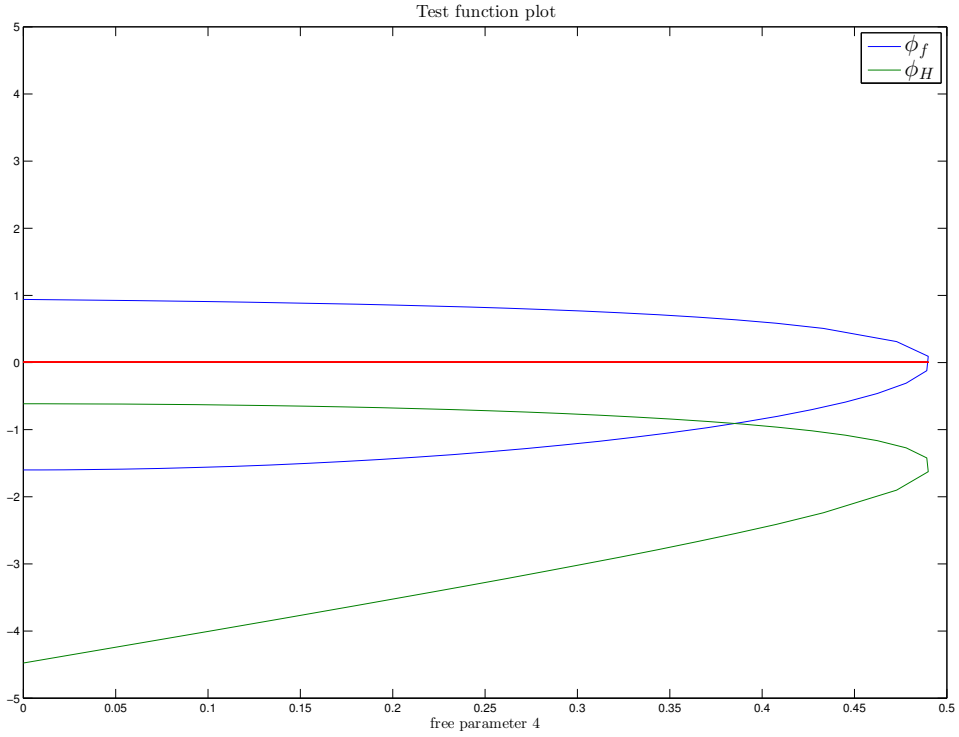

Figure 13: Plot of test functions for stst_br

```
BR_BIFDET: Bogdanov-Takens detected near par(2) = 0.0199989457, par(4) = 0.4835320970.
Failed to correct Bogdanov-Takens point.

BR_BIFDET: Bogdanov-Takens detected near par(2) = 0.0399981183, par(4) = 0.4757164567.
Failed to correct Bogdanov-Takens point.

BR_BIFDET: Cusp detected near par(2) = 1.3179630819, par(4) = 0.0462438690.
Failed to correct cusp.

BR_BIFDET: Bogdanov-Takens detected near par(2) = 1.3179630819, par(4) = 0.0462438690.
BR_BIFDET: a = -0.2307473313, b = -0.9154839442, par(2) = 1.3000000000, par(4) = 0.0505079212.

BR_BIFDET: Cusp detected near par(2) = 1.5990696079, par(4) = -0.0000094571.
BR_BIFDET: Cusp located at  par(2) = 1.6000000000, par(4) = -0.0000000000.
BR_BIFDET: Normal form coefficients: b = -0.0000000002, c = 0.5555557861.

BR_BIFDET: Bogdanov-Takens detected near par(2) = 1.5811766051, par(4) = -0.0008560336.
Failed to correct Bogdanov-Takens point.

BR_BIFDET: Cusp detected near par(2) = 1.2852191807, par(4) = -0.0540913384.
Failed to correct cusp.

BR_BIFDET: Bogdanov-Takens detected near par(2) = 1.2852191807, par(4) = -0.0540913384.
BR_BIFDET: a = 0.2353902067, b = 0.9123599189, par(2) = 1.3000000000, par(4) = -0.0505079212.
```

In Figure 15 we have plotted the test functions. The detection of a cusp point near a Bogdanov-Takens point can be explained. Indeed, the test function for the Bogdanov-Takens point $\psi^f_{BT} = p^T\Delta'(0)q$ is normalized to 1 in the calculation for the critical normal form of the

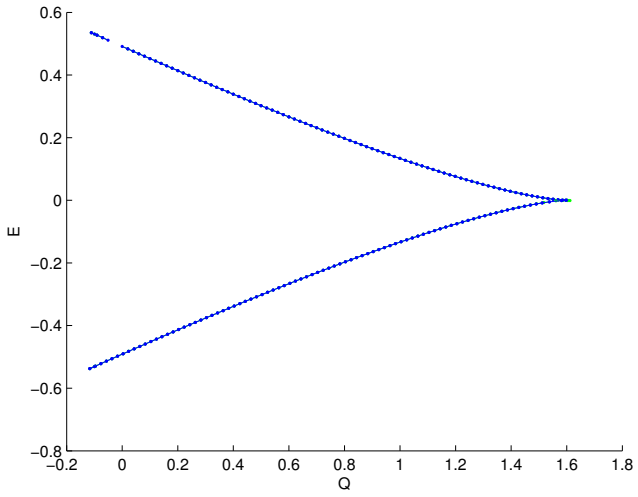

Figure 14: Plot the fold branch. Here we see a cusp bifurcation at $(Q, E) = (1.6, 0)$.

fold point. This is done by scaling the vector p. Near a Bogdanov-Takens point $p^T\Delta'(0)q$ becomes really small, which means the length of p becomes very large in order to achieve the normalization $p^T\Delta'(0)q$=1 . As a result the length of the critical normal form coefficient b in (9) of a fold point becomes very large.

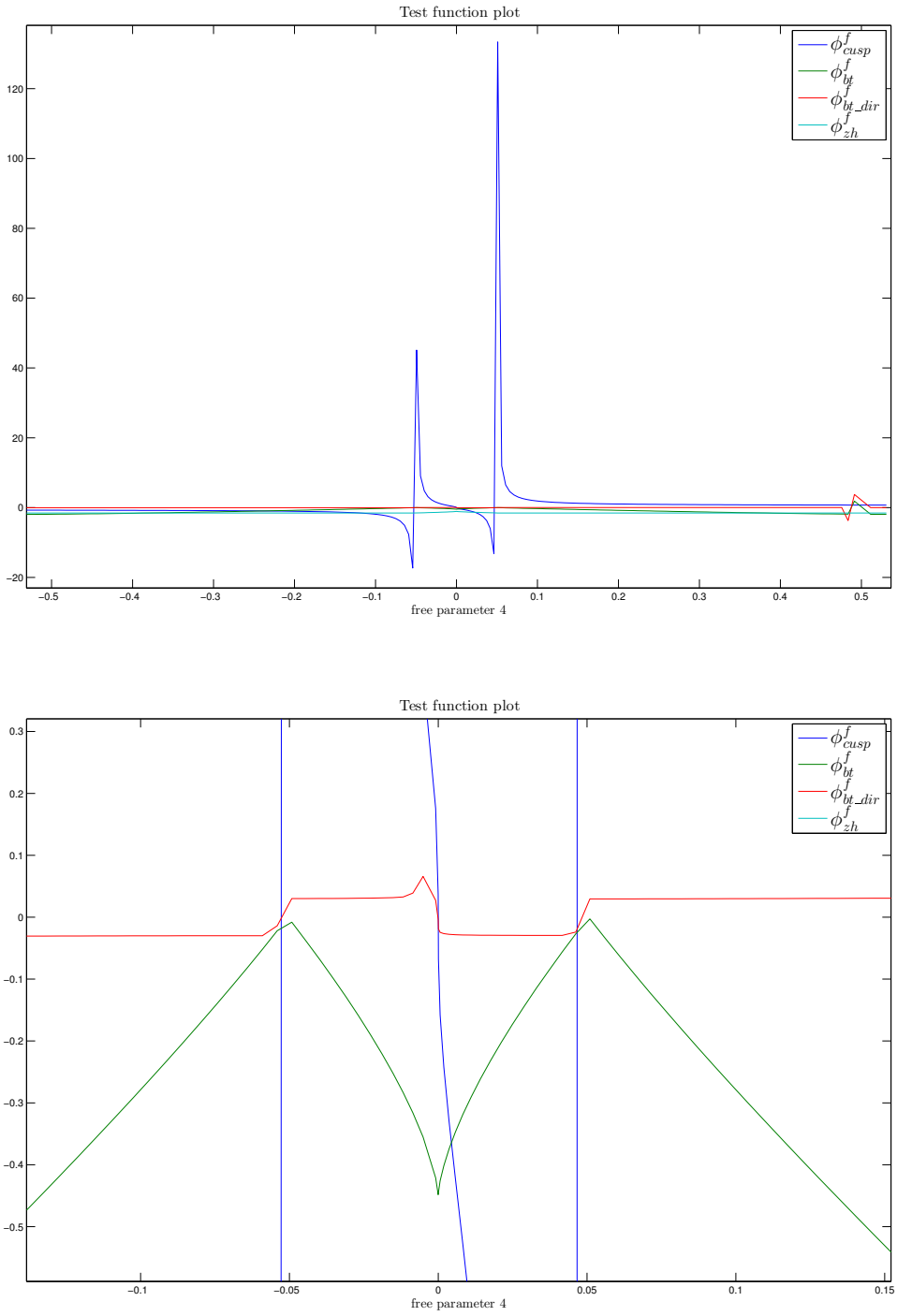

Figure 15: Plot of test functions for fold_br

References

[Diekmann et al., 2012] Diekmann, O., van Gils, S., Lunel, S., and Walther, H. (2012). *Delay Equations: Functional-, Complex-, and Nonlinear Analysis*. Applied Mathematical Sciences. Springer New York.

[Giannakopoulos and Zapp, 2001] Giannakopoulos, F. and Zapp, A. (2001). Bifurcations in a planar system of differential delay equations modeling neural activity. *Physica D: Nonlinear Phenomena*, 159(3):215–232.

[Janssens, 2007] Janssens, S. (2007). On a normalization technique for codimension two bifurcations of equilibria of delay differential equations. Master's thesis, Universiteit Utrecht, the Netherlands. Supervised by Yu.A. Kuznetsov.

[Kuznetsov, 2004] Kuznetsov, Yu. A. (2004). *Elements of Applied Bifurcation Theory*. Springer-Verlag.

[Liu et al., 2013] Liu, X., Liu, Y., and Wang, J. (2013). Bogdanov-Takens bifurcation of a delayed ratio-dependent Holling-Tanner predator prey system. In *Abstract and Applied Analysis*, volume 2013. Hindawi Publishing Corporation.

[Visser, 2013] Visser, S. (2013). *From spiking neurons to brain waves*. PhD thesis, University of Twente, Enschede.

[Visser et al., 2012] Visser, S., Meijer, H. G., van Putten, M. J., and van Gils, S. A. (2012). Analysis of stability and bifurcations of fixed points and periodic solutions of a lumped model of neocortex with two delays. *The Journal of Mathematical Neuroscience (JMN)*, 2(1):1–24.

[Wage, 2014] Wage, B. (2014). Normal form computations for delay differential equations in DDE-BIFTOOL. Master's thesis, Utrecht University. Supervised by Yu.A. Kuznetsov, available at `http://dspace.library.uu.nl/handle/1874/296912`.