Interplay between normal forms and center manifold reduction for homoclinic predictors near Bogdanov-Takens bifurcation

Maikel Bosschaert

Jul 17, 2022

CONTENTS

In this Jupyter Book we will demonstrate the effectiveness of the homoclinic predictors derived in [Kuz21] to start continuations of homoclinic orbits in two parameters in MatCont near generic codimension two Bogdanov-Takens bifurcation points in n-dimensional ordinary differential equations (ODEs) of the form

$$\dot{x}(t) = f(x(t), \alpha), \tag{1}$$

where $x : \mathbb{R} \to \mathbb{R}^n$ and $\alpha \in \mathbb{R}^m$. The function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is assumed to be as smooth as necessary.

In total there are eight different models considered. Each the models consists of two notebooks:

- one to generate the necessary *derivative files*, see for example *Generate system files for Morris-Lecar in a platinum model*,
- and one *model notebook* to analyse the model.

In the first two models *Morris-Lecar Model*, and *CO-oxidation in a platinum model.* we partly follow the presentation in [BAH15], that is, we define an equilibrium of (1), continue the equilibrium point one parameter, encounter a codimension one Hopf or fold bifurcation, continue the codimension one bifurcation point in two parameters and encounter one or more Bogdanov-Takens points. The homoclinic orbits emanating from these Bogdanov-Takens point can then be continued using one of the homoclinic predictors form [Kuz21] implemented into MatCont. In these first two models we will describe in detail how a homoclinic approximation near the generic codimension two Bogdanov-Takens bifurcation is obtained in MatCont.

Although the procedure described above is a common situation for discovering Bogdanov-Takens points in ODEs, in many situations it is possible to derive the Bogdanov-Takens point directly, either analytically or numerically, form the ODE. This will be the approach for the remaining models were we will focus solely on the homoclinic orbits emanating from the Bogdanov-Takens point and not the Hopf and fold curves.

At the end of each *model notebook* we create a convergence plot of the different methods to approximation the homoclinic solution derived in [Kuz21].

In the notebook *BogdanovTakens.ipynb* the importance of using a higher order approximation of the non-linear time transformation used in [Kuz21] is demonstrated.

Note: The *derivative files* can also be generated with the graphical user interface of MatCont. In fact, if preferred, the continuation of the various bifurcation branches can be done using only the graphical user interface.

Part I

Morris-Lecar

CHAPTER

ONE

GENERATE SYSTEM FILES FOR MORRIS-LECAR IN A PLATINUM MODEL

In this script the **system files** for the Morris-Lecar model [ML81]

$$\begin{cases} c\dot{V} = I_{app} - I_{ion}, \\ \dot{w} = \phi \dfrac{w_\infty - w}{\tau}, \end{cases}$$

where

$$\begin{aligned} m_\infty &= 0.5\left(1 + \tanh\left(\frac{v - v_1}{v_2}\right)\right), \\ w_\infty &= 0.5\left(1 + \tanh\left(\frac{v - v_3}{v_4}\right)\right), \\ i_{ion} &= g_{ca} m_\infty (v - v_{ca}) + g_k w (v - v_k) + g_l (v - v_l), \\ \tau &= \operatorname{sech}\left(\frac{v - v_3}{2 v_4}\right). \end{aligned}$$

are generated. These are used in the *Morris-Lecar demo*.

1.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

1.2 Set the system name

```
system_name = 'Morris_Lecar';
```

1.3 Create coordinates and parameter names as strings

```
coordsnames = {'V', 'w'};
parnames = {'Iapp', 'v3'};
```

1.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `alpha` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});       % create symbol for alpha and delta
par=cell2sym(parnames);  % now alpha1 is par(1) etc
syms(coordsnames{:});    % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

1.5 Define fixed parameters

```
C=20;
VL=-60;
VCa=120;
VK=-84;
gL=2;
gCa=4.4;
gK=8;
v1=-1.2;
v2=18;
v4=30;
phi=1/25;
```

1.6 Define the system

```
minf = 0.5*(1+tanh((V-v1)/v2));
winf = 0.5*(1+tanh((V-v3)/v4));
Iion = gCa*minf*(V-VCa) + gK*w*(V-VK) + gL*(V-VL);
tau = sech((V-v3)/2/v4);
dV_dt = (Iapp - Iion)/C;
dw_dt = phi/tau*(winf-w);
system = [dV_dt; dw_dt];
```

In general there are no modifications needed after this line.

1.7 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

1.8 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

TWO

MORRIS-LECAR MODEL

In [ML81] the following reduced model is considered

$$\begin{cases} c\dot{V} = I_{app} - I_{ion}, \\ \dot{w} = \phi \dfrac{w_\infty - w}{\tau}, \end{cases}$$

where

$$\begin{aligned} m_\infty &= 0.5\left(1 + \tanh\left(\frac{v - v_1}{v_2}\right)\right), \\ w_\infty &= 0.5\left(1 + \tanh\left(\frac{v - v_3}{v_4}\right)\right), \\ i_{ion} &= g_{ca} m_\infty (v - v_{ca}) + g_k w (v - v_k) + g_l (v - v_l), \\ \tau &= \operatorname{sech}\left(\frac{v - v_3}{2v_4}\right). \end{aligned}$$

are generated. Here, C is the membrane capacitance; I_{app} is the applied current; I_{ion} collects the Ca^{2+}, K^{1+} and L currents; g_L, g_{Ca} and g_K are the maximal conductances for L, Ca^{2+} and K^{1+} channels, respectively. V is the membrane potential; V_L, V_{Ca} and V_K are the equilibrium potentials corresponding to L, Ca^{2+} and K^{1+} conductance's, respectively; w is the fraction of open K^{1+} channels; m_∞ and w_∞ are the fractions of open Ca^{2+} and K^{1+} channels at steady state respectively; $\frac{\tau}{0}$ determines the activation time for the K^{1+} current. The parameters v_1, v_2, v_3 and v_4 are chosen to fit the model data.

The following parameters are fixed

$$C = 20, \quad V_L = -60, \quad V_{Ca} = 120, \quad V_K = -84, \quad g_L = 2, \quad g_{Ca} = 4.4, \quad g_K = 8,$$

and

$$v_1 = -1.2, \quad v_2 = 18, \quad v_3 = 2, \quad v_4 = 30, \quad \phi = \frac{1}{25}.$$

2.1 Overview

In this demo we will revisit and extend the results from [BAH15], using the new homoclinic predictor from [Kuz21]. Thus, we will:

- Compute a curve of equilibria, parametrized by I_{app}.
- Detect two Hopf points.
- Start continuation from the first Hopf point in two parameters (I_{app}, v_3).

- Detect four Bogdanov-Takens points.
- Start continuation from the Bogdanov-Takens points in two parameters (I_{app}, v_3).
- Compare the predicted and computed homoclinic bifurcation curve emanating from the first the Bogdanov-Takens point in parameters space.
- Compare a range of predictors for the homoclinic solutions emanating from the first Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space.
- Create bifurcation plots including Hopf and fold curves.
- Create a convergence plot comparing the different homoclinic approximations derived in [Kuz21].

2.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Equilibrium'])
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
```

2.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *Morris-LecarGenSym.ipynb*.

```
odefile=@Morris_Lecar;
```

2.4 Define equilibrium

We manually define an equilibrium at

$$(V, w) = (-60.85456779, 0.0149139691), \tag{1}$$

with parameter values $I_{app} = 0$ and $v_3 = 2$.

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in the software package *DDE-BifTool* [SEL+14].

```
parnames = {'Iapp', 'v3'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
p(ind.Iapp) = 0;
p(ind.v3) = 2;
x  = [-60.85456779; 0.0149139691];
```

2.5 Continue equilibrium in parameter I_{app}

To continue the equilibrium (1) in parameter I_{app}, we first need to obtain a tangent vector to the curve. This is done by the function `init_EP_EP`. Then we use the function `contset` to obain a **struct** containing a list of options which is passed on to the continuer. By adjusting the values of the fields of the `opt` **struct** we set the maximum step size. We also set the maximum number of points to continue and weather or not to detect bifurcation points (`opt.Singularities`) on the equilibrium curve. For more information about all options available to the *MatCont* continuer and the continuation process in general, we refer to [DGK+08].

Finally, we continue the curve using the function `cont`.

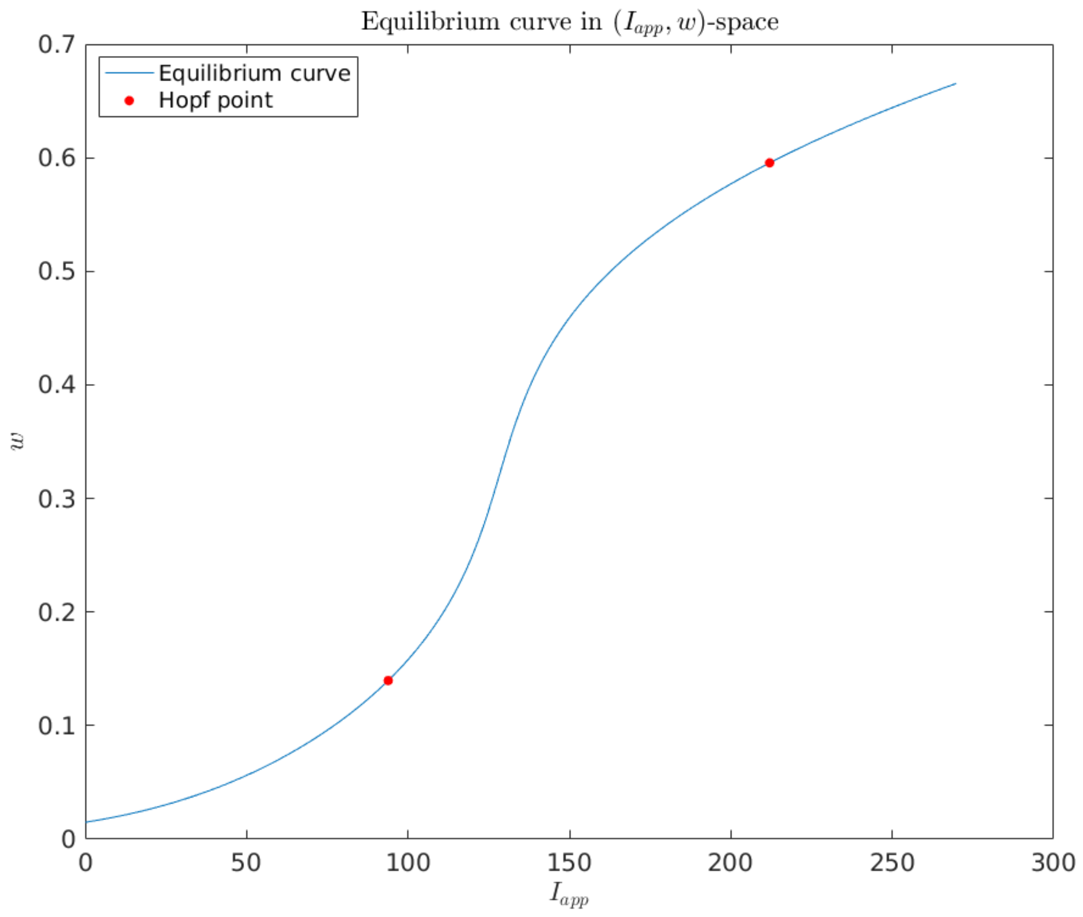
```
[x1_pred, v1_pred] = init_EP_EP(odefile, x, p, ind.Iapp);
opt = contset;
opt.MaxStepsize    = 1;
opt.MaxNumPoints   = 300;
opt.Singularities = 1;
[eqbr_x, ~, eqbr_bif_data] = cont(@equilibrium, x1_pred, v1_pred, opt);
```

```
first point found
tangent vector to first point found
label = H , x = ( -25.270107 0.139673 93.857614 )
First Lyapunov coefficient = 5.220142e-04
label = H , x = ( 7.800664 0.595491 212.018815 )
First Lyapunov coefficient = 5.451161e-04

elapsed time  = 0.2 secs
npoints curve = 300
```

There are two Hopf bifurcation points detected (H). The **array struct** `eqbr_bif_data` contains information about the detected bifurcation points. We use this to extract the index of the detected bifurcation points on the equilibrium curve `eqbr_x`. The equilibrium curve `eqbr_x` is just a two dimensional array. Each column consists of a point on the curve. The first two rows contain the point (V, w) while the last row contains the parameter I_{app}.

Below we plot the equilibrium curve `eqbr_x`, together with the detected Hopf points, in (I_{app}, w)-space.

```
%plot --width 1024 --height 800
hopfPointsInfo   = eqbr_bif_data(strcmp({eqbr_bif_data.msg}, 'Hopf')==1);
HopfPoint1 = eqbr_x(:,hopfPointsInfo(1).index);
HopfPoint2 = eqbr_x(:,hopfPointsInfo(2).index);
plot(eqbr_x(3,:), eqbr_x(2,:)); hold on
plot(HopfPoint1(3), HopfPoint1(2), '.r' ,'MarkerSize', 20)
plot(HopfPoint2(3), HopfPoint2(2), '.r' ,'MarkerSize', 20)
xlabel('$I_{app}$')
ylabel('$w$')
legend({'Equilibrium curve', 'Hopf point'}, 'Location', 'NorthWest')
title('Equilibrium curve in $(I_{app},w)$-space')
```



2.6 Setup Hopf point

To continue the first Hopf point in the parameters I_{app} and v_3 we construct a new point `hopf1` containing the position and parameter values. These are needed to obtain an initial tangent vector - using the function `init_H_H` - in the full phase/parameter space. Since, from now on, we will be using the continuation parameters I_{app} and v_3 frequently we assigned these parameters to the variable `ap` (active parameters).

```
ap = [ind.Iapp ind.v3]; % continuation parameters
hopf1.par = p;
hopf1.par(ap) = eqbr_x(3, hopfPointsInfo(1).index);
hopf1.x = eqbr_x(1:2, hopfPointsInfo(1).index);
[x1, v1] = init_H_H(odefile, hopf1.x, hopf1.par', ap);
```

2.7 Continue Hopf point in parameters I_{app} and v_3

We continue the Hopf point curve using again the function `cont`. We use the same continuation options as before defined above in the **struct** `opt`, but set additionally the following options. We increase the number of maximum allowed continuation points. We also increase the accuracy for locating detected bifurcations (`TestTolerance`) and the maximum number of iterations that may be used to achieve this (`MaxTestIters`). This improves the homoclinic predictor which depend directly on the accuracy of the located Bogdanov-Takens point.

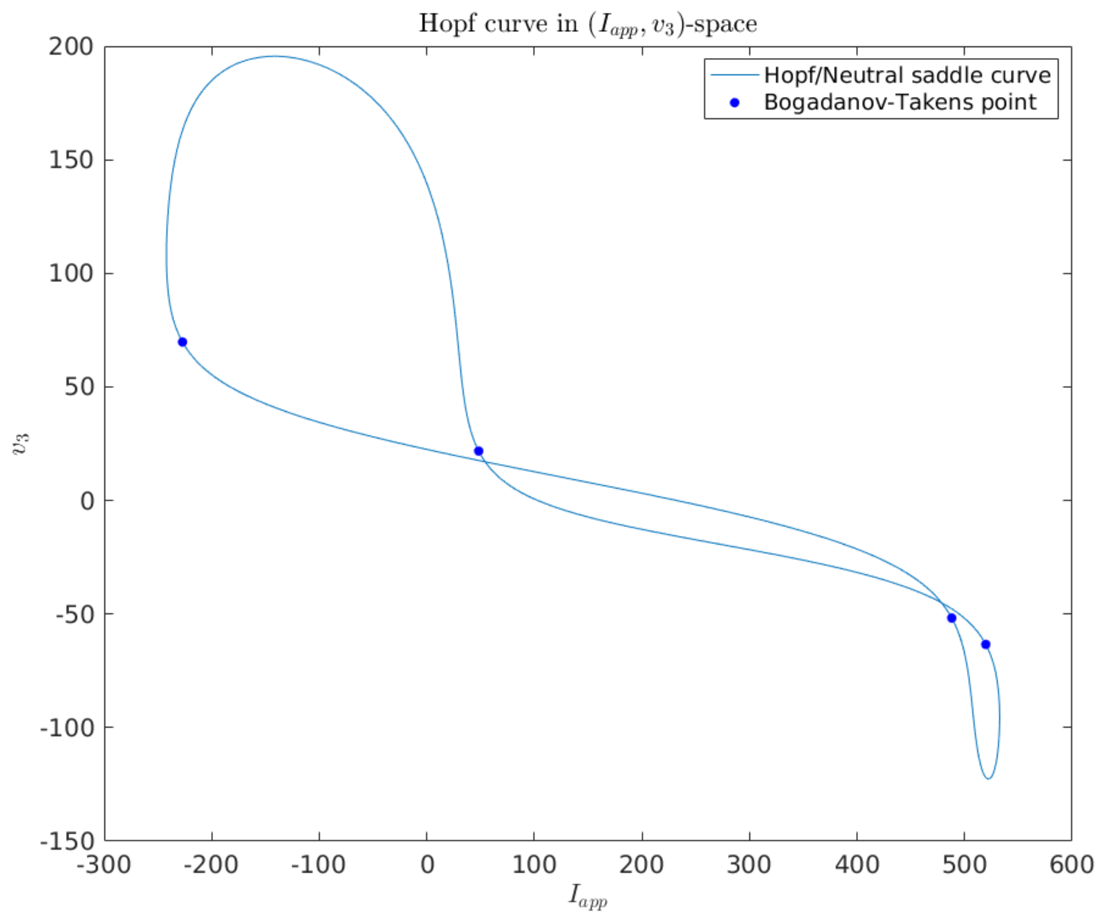
```
opt.TestTolerance = 1e-15;
opt.MaxTestIters = 10;
opt.MaxNumPoints = 2000;
[hopf_br, ~, hopf_br_bif] = cont(@hopf, x1, v1, opt);
```

```
first point found
tangent vector to first point found
label = BT, x = ( 13.611147 0.023136 -227.131888 69.755737 0.000000 )
(a,b)=(-1.119908e-03, -3.301195e-02)
label = BT, x = ( 2.815616 0.974408 487.997703 -51.777429 0.000000 )
(a,b)=(-7.485863e-04, -2.961373e-02)
label = BT, x = ( -11.152829 0.970208 519.625365 -63.401902 0.000000 )
(a,b)=(7.383698e-04, 2.289394e-02)
label = BT, x = ( -27.149904 0.037007 48.225939 21.734143 0.000000 )
(a,b)=(2.766145e-04, 1.513072e-02)
Closed curve detected at step 1866

elapsed time  = 2.3 secs
npoints curve = 1866
```

There are four Bogdanov-Takens bifurcation points (BT) detected on the Hopf branch `hopf_br`.

As with the Hopf points, information about the detected bifurcation points is stored in the **struct array** `hopf_br_bif`. Below we extract the Bogdanov-Takens bifurcation points.

```
bt_points_info = hopf_br_bif(strcmp({hopf_br_bif.label}, 'BT')==1);
BTPoint1 = hopf_br(:,bt_points_info(1).index);
BTPoint2 = hopf_br(:,bt_points_info(2).index);
BTPoint3 = hopf_br(:,bt_points_info(3).index);
BTPoint4 = hopf_br(:,bt_points_info(4).index);
plot(hopf_br(3,:), hopf_br(4,:)); hold on
plot(BTPoint1(3), BTPoint1(4), '.b' ,'MarkerSize', 20)
plot(BTPoint2(3), BTPoint2(4), '.b' ,'MarkerSize', 20)
plot(BTPoint3(3), BTPoint3(4), '.b' ,'MarkerSize', 20)
plot(BTPoint4(3), BTPoint4(4), '.b' ,'MarkerSize', 20)
xlabel('$I_{app}$')
ylabel('$v_3$')
legend({'Hopf/Neutral saddle curve', 'Bogadanov-Takens point'}, 'Location', 'NorthEast
↪')
title('Hopf curve in $(I_{app},v_3)$-space')
```



2.8 Initial prediction of homoclinic orbit near Bogdanov-Takens point 1

To obtain an initial approximation to the homoclinic solution near the Bogdanov-Takens point we use the function `init_BT_Hom`. Its arguments are the system file (`odefile`), the Bogdanov-Takens point (`bt1`) as defined below, the unfolding parameters (`ap`) and an options structure (`BToptions`). The options structure created with the function `BT_Hom_set_options` contains the following fields:

- `ntst` Number of mesh intervals with a default value of 40.
- `ncol` Number of collocation points used in each interval with a default of 4.
- `extravec` Three dimensional boolean row vector indicating which *homoclinic parameters* are selected to be free. The first component refers to the half-return time, while the second and third components refer to the distances from the saddle point to the first, respectively, the last point on the homoclinic orbit. The default value is set to `[0 1 1]`. Thus, the half-return time `T` is fixed.
- `order` The order of the homoclinic approximation used with a default value of 3.
- `amplitude` Desired amplitude of the homoclinic solution. If left empty then a conservative estimate is made, see [Kuz21].

- `TTolerance` Desired distance between the last point on the numerical homoclinic solution and the saddle point. This should be at least be smaller than the amplitude. If left empty it is defined by `amplitude*1.0e-03`.
- `HigherOrderTimeReparametrization` Boolean to indicate if a higher order approximation to the non-linear time transformation in the Lindstedt-Poincaré method should be used. This should always be set to `1`. It is only implemented for demonstration purposes.
- `method` Selects the method to be used to approximate the homoclinic solution. The different methods available are:
  - orbital (the default),
  - orbitalv2,
  - LP (Lindstedt-Poincaré with smooth normal form),
  - LPHypernormalForm,
  - RegularPerturbation,
  - RegularPerturbationL2.

  We refer to [Kuz21] for the interpretations.
- `messages` Boolean to indicate if information about selected parameter should be printed the console. The default value is set to `true`.
- `correct` Boolean to indicate if the predicted homoclinic solution should be corrected with Newton. The default value is set to `true`.

Here we will use the default values for the Bogdanov-Takens option structure, except we set the field `correct` to `false` and manually correct the approximation.

```
bt_index = bt_points_info(1).index;
bt1.x = hopf_br(1:2, bt_index);
bt1.par = p';
bt1.par(ap) = hopf_br(3:4, bt_index);
BToptions = BT_Hom_set_options();
BToptions.correct = false;
[x1_pred, v1_pred] = init_BT_Hom(odefile, bt1, ap, BToptions);
```

```
Center manifold coefficients' accuracy: 2.842171e-14
BT normal form coefficients:
a=-1.119908e-03,          b=-3.301195e-02
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.0616583
The initial half-return time T: 1221.71
The initial distance eps0: 3.339e-05
The initial distance eps1: 0.000109
```

2.9 Correct initial prediction of homoclinic orbit near bt1 with Newton

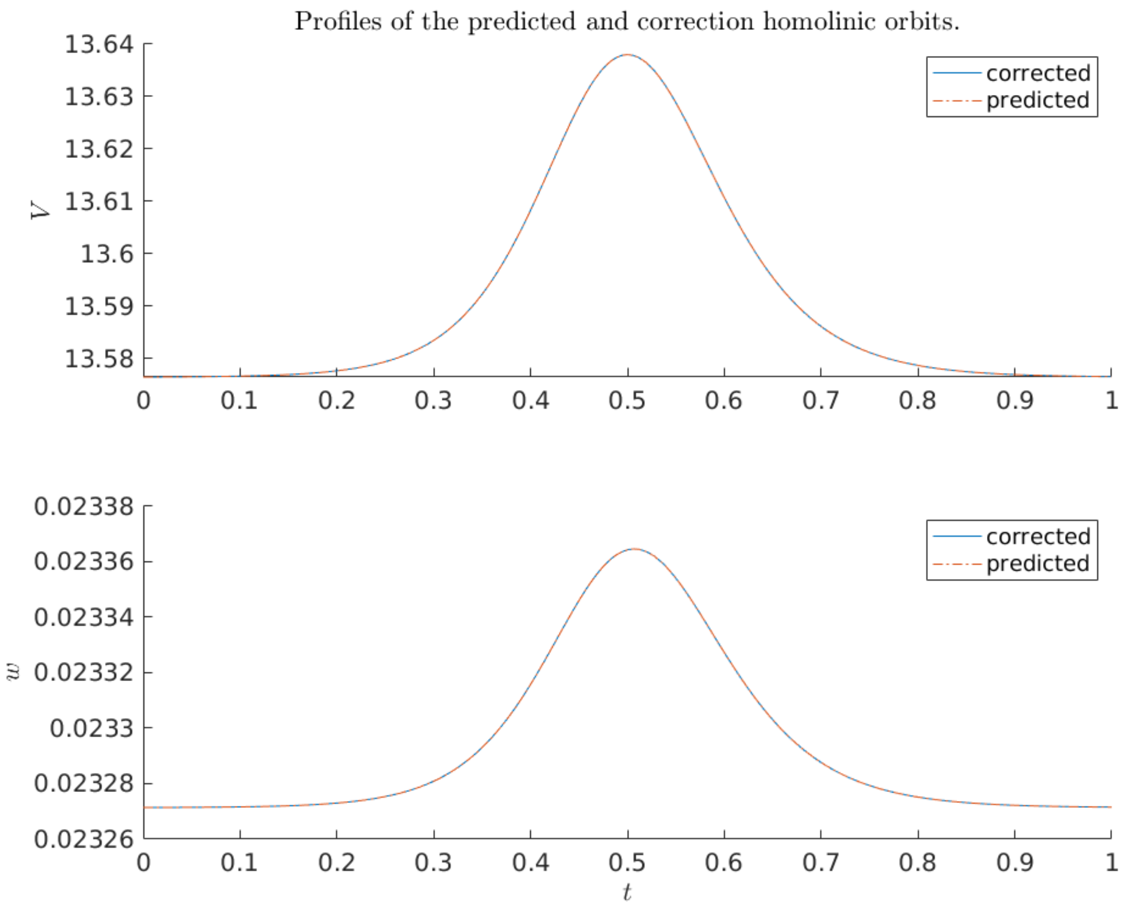
Now that we have an initial prediction for the homoclinic orbit we manually correct it using Newton. After the homoclinic predictor is corrected with the **MatCont** function `newtcorr` we use the function `bt_rearr` (Bogdanov-Takens rearrange) to extract the homoclinic orbit and saddle point from the homoclinic correction.

```
[hom1_x, hom1_v, ~] = newtcorr(x1_pred, v1_pred);
[x1_orbit, x1_saddle] = bt_rearr(hom1_x);
```

2.10 Compare profiles of predicted and corrected solution (bt1)

Using again the **MatCont** function `bt_rearr`, but now on the homoclinic prediction `x1_pred` we compare the profiles of the predicted and corrected homoclinic orbits. We see that they are indistinguishable. Note that to access the mesh on which the homoclinic orbit is computed we need the global variable `homds`.

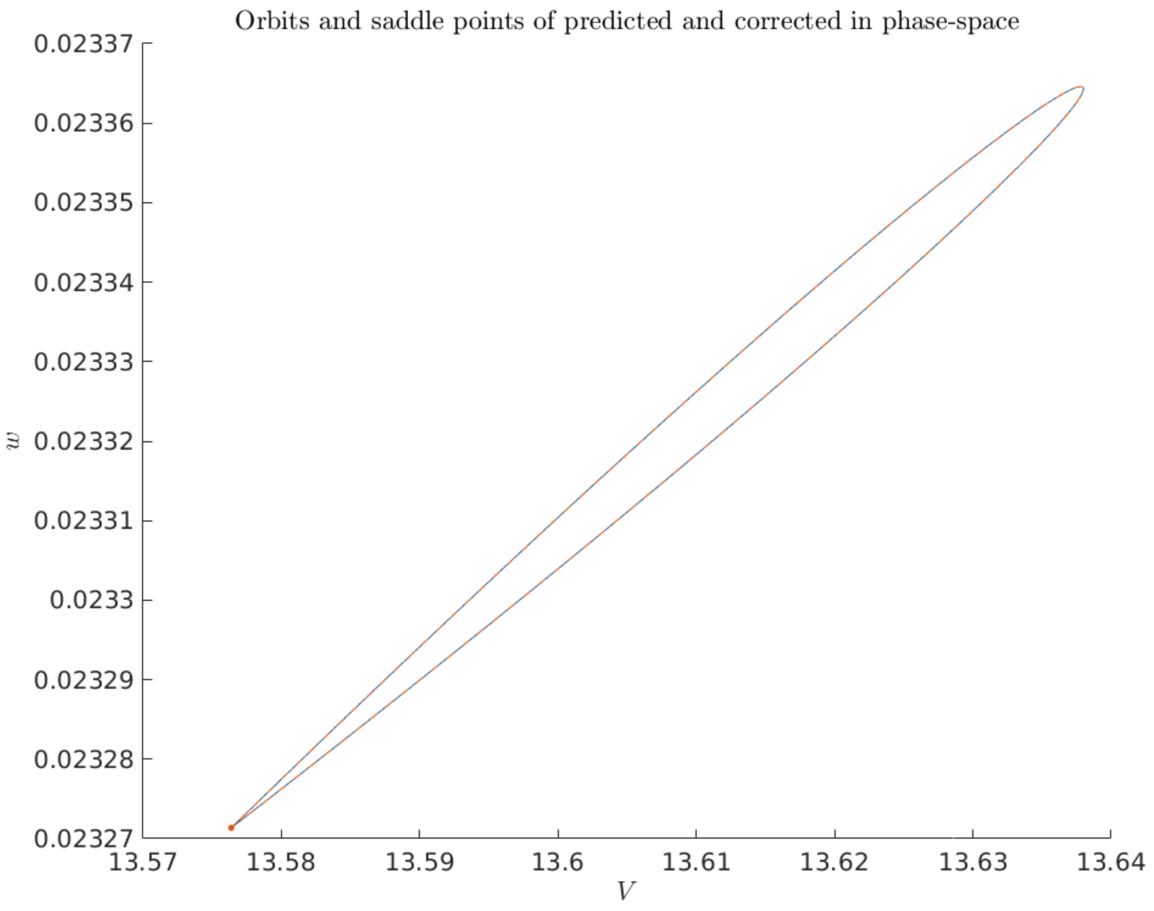
```
[homoclinic1_pred, saddle1_pred] = bt_rearr(x1_pred);
subplot(2,1,1); hold on;
global homds
title('Profiles of the predicted and correction homolinic orbits.')
plot(homds.finemsh, x1_orbit(1:2:end))
plot(homds.finemsh, homoclinic1_pred(1:2:end),'-.')
legend({'corrected', 'predicted'})
ylabel('$V$')
subplot(2,1,2); hold on;
plot(homds.finemsh, x1_orbit(2:2:end))
plot(homds.finemsh, homoclinic1_pred(2:2:end),'-.')
legend({'corrected','predicted'})
ylabel('$w$')
legend({'corrected',  'predicted'})
xlabel('$t$')
```



2.11 Compare predictor and corrected solution in (V, w) phase-space

Below we compare the predicted and corrected homoclinic orbit in (V, w) phase-space, as well as the predicted and corrected saddle point.

```
hold on
plot(x1_orbit(1:2:end),x1_orbit(2:2:end))
plot(homoclinic1_pred(1:2:end),homoclinic1_pred(2:2:end),'-.')
plot(x1_saddle(1), x1_saddle(2),'.', 'MarkerSize', 12, 'Color', [0 0.4470 0.7410])
plot(saddle1_pred(1), saddle1_pred(2),'.', 'MarkerSize', 12, 'Color', [0.8500, 0.3250,
 ↪ 0.0980])
xlabel('$V$')
ylabel('$w$')
title('Orbits and saddle points of predicted and corrected in phase-space')
```



2.12 Continue homoclinic curve emanating from bt1

Having obtain an initial approximation `[hom_x, hom_v]`, where `homo_v` is the tangent vector to the homoclinic curve pointing outwards from the Bogdanov-Takens point, we can start continuation using the function `cont`. Since, we are not interested in bifurcations of the homoclinic orbit, we disable the detection of homoclinic bifurcations `opt.Singularities = 0`. This will reduce the computational cost. We also set the maximum number of continuation steps.

```
opt.Singularities = 0;
opt.MaxNumPoints = 1000;
homoclinic_br1 = cont(@homoclinic, hom1_x, hom1_v, opt);
```

```
first point found
tangent vector to first point found
Current step size too small (point 477)
elapsed time  = 18.7 secs
npoints curve = 477
```

2.13 Compare predicted with computed parameters emanating from bt1

Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm_orbital` to obtain the smooth orbital normal form coefficients, i.e. a and b, and the coefficients of the transformation K between the parameters of the system and the parameters in the smooth orbital normal form, see [Kuz21].

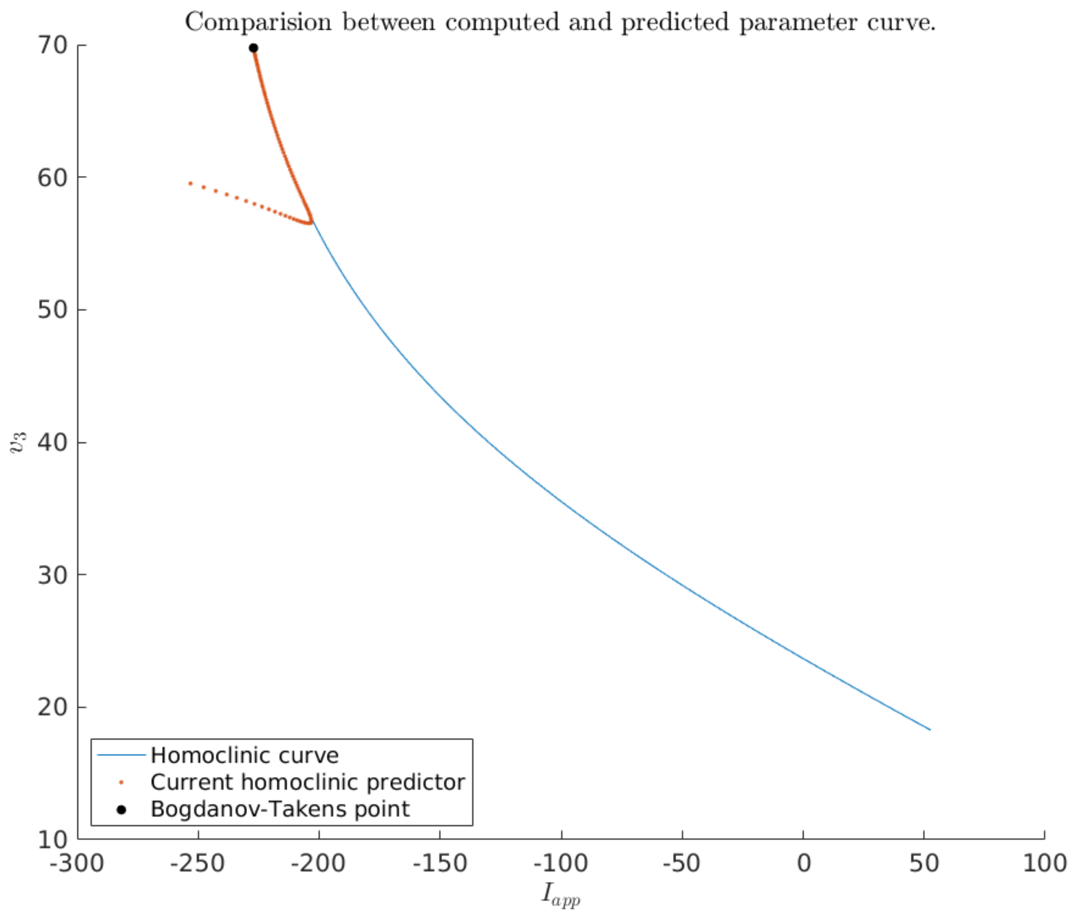
```
hold on
% plot computed homoclinic parameter curve
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent smooth orbital normal form coefficients
bt1 = BT_nmfm_orbital(odefile, bt1, ap, BToptions);
a   = bt1.nmfm.a;
b   = bt1.nmfm.b;
K10 = bt1.nmfm.K10;
K01 = bt1.nmfm.K01;
K02 = bt1.nmfm.K02;
K11 = bt1.nmfm.K11;
K03 = bt1.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 1.8);
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alpha = bt1.par(ap) + alpha;
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.')
% plot Bogdanov-Takens point
plot(bt1.par(ind.Iapp), bt1.par(ind.v3), '.k', 'MarkerSize', 20)
% set axis labels and legend
```

(continues on next page)

(continued from previous page)

```
xlabel('$I_{app}$')
ylabel('$v_3$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'SouthWest')
title('Comparision between computed and predicted parameter curve.')
```

```
Center manifold coefficients' accuracy: 2.842171e-14
```



2.14 Bifurcation diagram in (V, w) phase-space

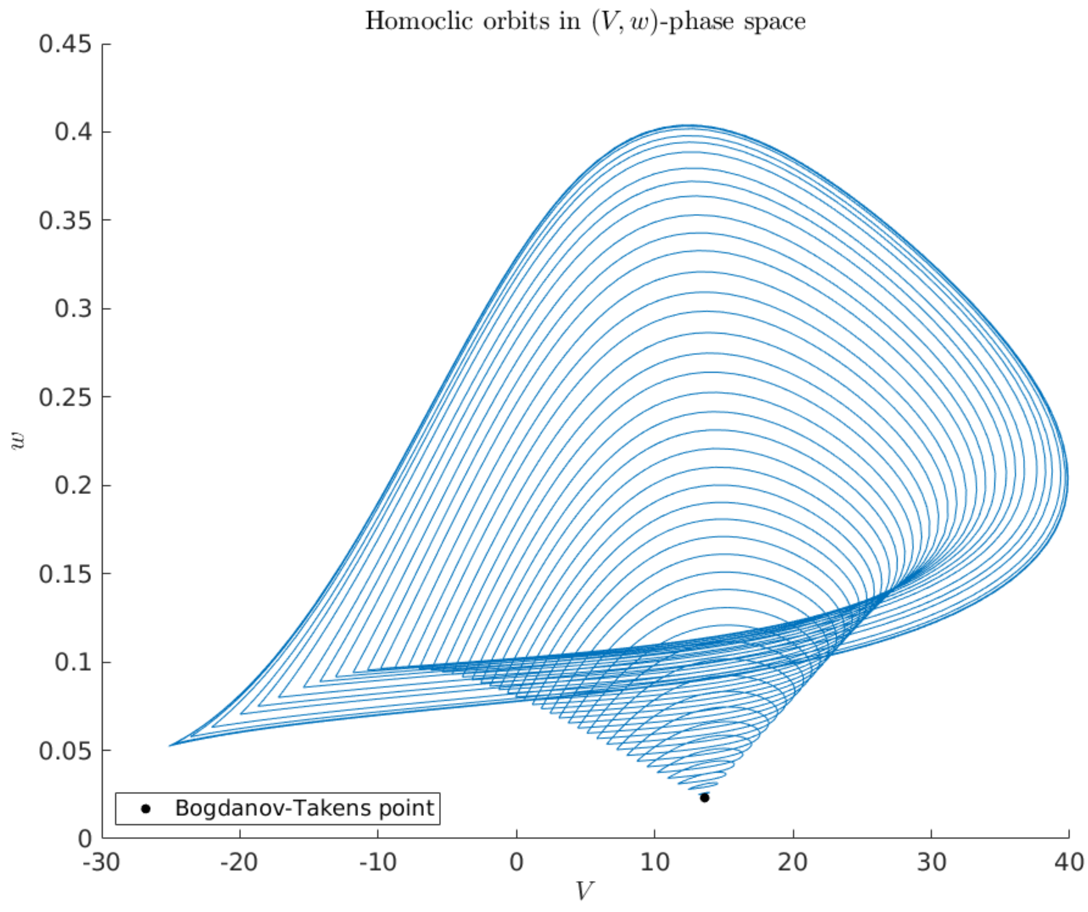
To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (V, w) phase-space.

```
hold on
plot(homoclinic_br1(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br1(homds.coords(2:homds.nphase:end), 1:10:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$V$')
ylabel('$w$')
plot(bt1.x(1), bt1.x(2), '.k' ,'MarkerSize', 20)
```

(continues on next page)

(continued from previous page)

```
legend('Bogdanov-Takens point', 'Location', 'SouthWest')
title('Homoclic orbits in $(V,w)$-phase space')
```



2.14.1 Predictors of orbits for various epsilons

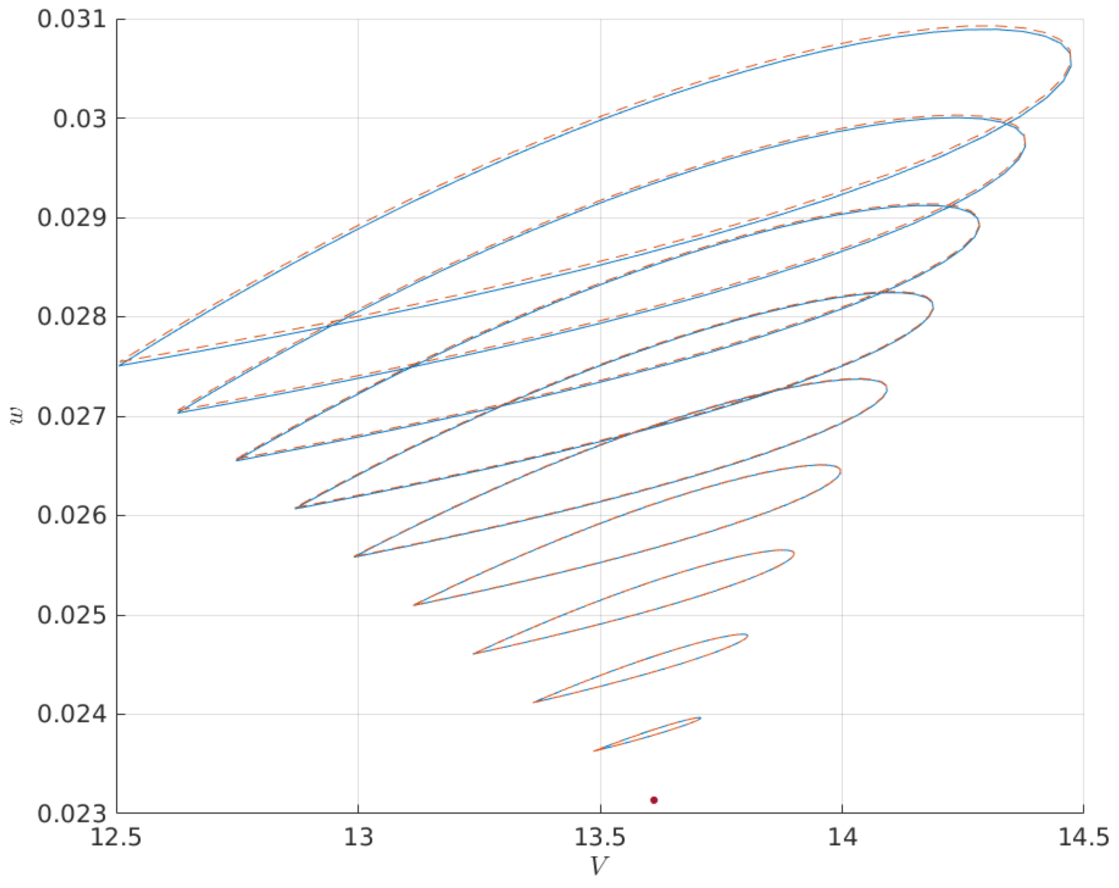
Before proceeding with continuing the homoclinic orbits emanating from the remaining three Bogdanov-Takens points we show that the estimate of the amplitude is very conservative. Below we compute for a large range of amplitudes the predicted and corrected homoclinic solutions and compare them in phase space. We see that for this model even with an amplitude of 2 the predicted homoclinic orbit is still very close.

```
options = BT_Hom_set_options();
options.messages = false;
options.correct = false;
options.TTolerance = 1.0e-05;

amplitudes = linspace(1.0e-03, 2,10);
XPredicted = zeros(330,length(amplitudes));
XCorrected = zeros(330,length(amplitudes));
for j=1:length(amplitudes)
  options.amplitude = amplitudes(j);
  [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, options);
```

(continues on next page)

(continued from previous page)

```
  XPredicted(:,j) = x_pred;
  try
    XCorrected(:,j) = newtcorr(x_pred, v0);
  catch
    warning('Didn''t convergence to homoclinic solution')
  end
end

hold on
cm = lines;
plot(XPredicted(homds.coords(1:homds.nphase:end),1:10), ...
     XPredicted(homds.coords(2:homds.nphase:end),1:10), ...
      'color', cm(1,:))
plot(XCorrected(homds.coords(1:homds.nphase:end),1:10), ...
     XCorrected(homds.coords(2:homds.nphase:end),1:10), ...
      '--', 'color', cm(2,:))
plot(bt1.x(1), bt1.x(2), '.', 'MarkerSize', 16)
xlabel('$V$')
ylabel('$w$')
grid on
```



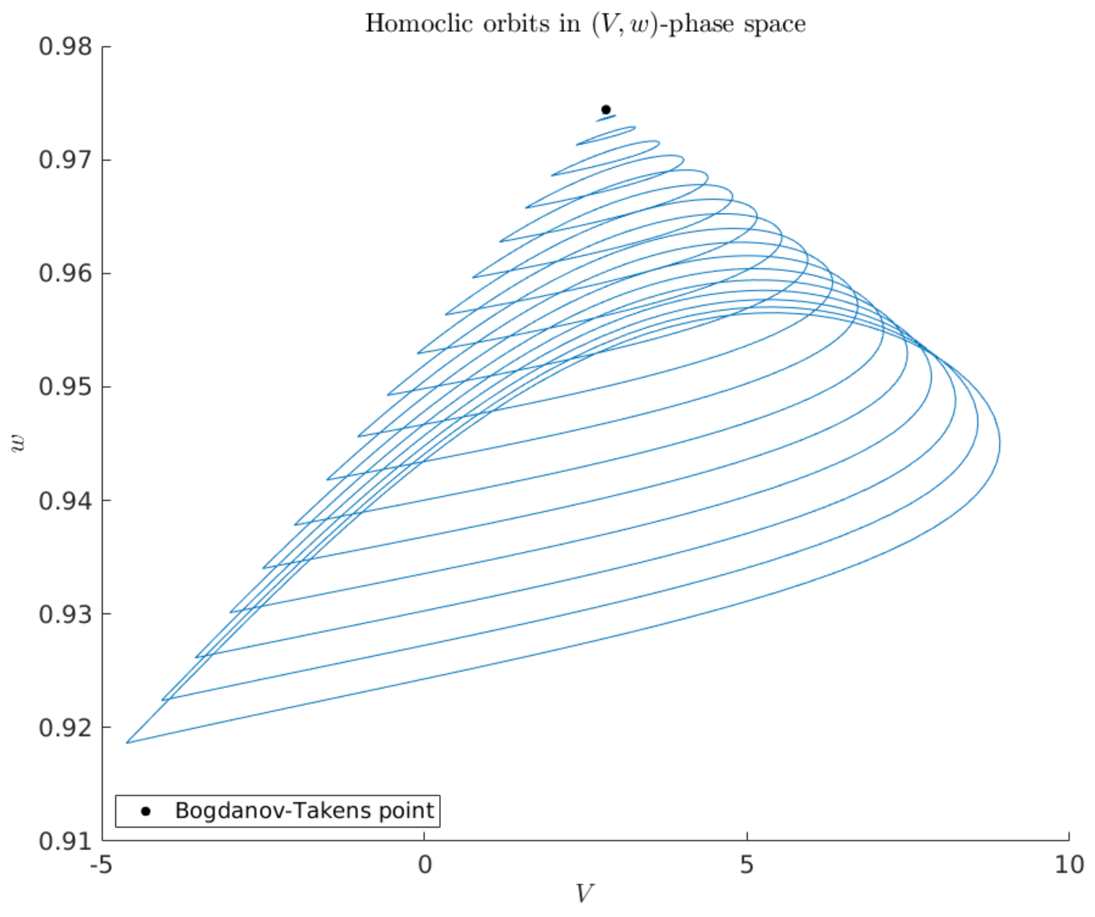
2.15 Continue homoclinic orbits from the remaining Bogdanov-Takens bifurcation points

The code below continues and plots the homoclinic orbits emanating for each of the remaining three detected Bogdanov-Takens points.

```
bt_index = bt_points_info(2).index;
bt2.x = hopf_br(1:2, bt_index);
bt2.par = p';
bt2.par(ap) = hopf_br(3:4, bt_index);
opt.MaxNumPoints = 100;
BToptions.correct = true;
[hom2_pred, hom2_v_pred] = init_BT_Hom(odefile, bt2, ap, BToptions);
homoclinic_br2 = cont(@homoclinic, hom2_pred, hom2_v_pred, opt);
```

```
Center manifold coefficients' accuracy: 3.552714e-15
BT normal form coefficients:
a=-7.485863e-04,         b=-2.961373e-02
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.0512162
The initial half-return time T: 1637.03
The initial distance eps0: 2.85469e-05
The initial distance eps1: 9.2748e-05
first point found
tangent vector to first point found

elapsed time  = 3.6 secs
npoints curve = 100
```

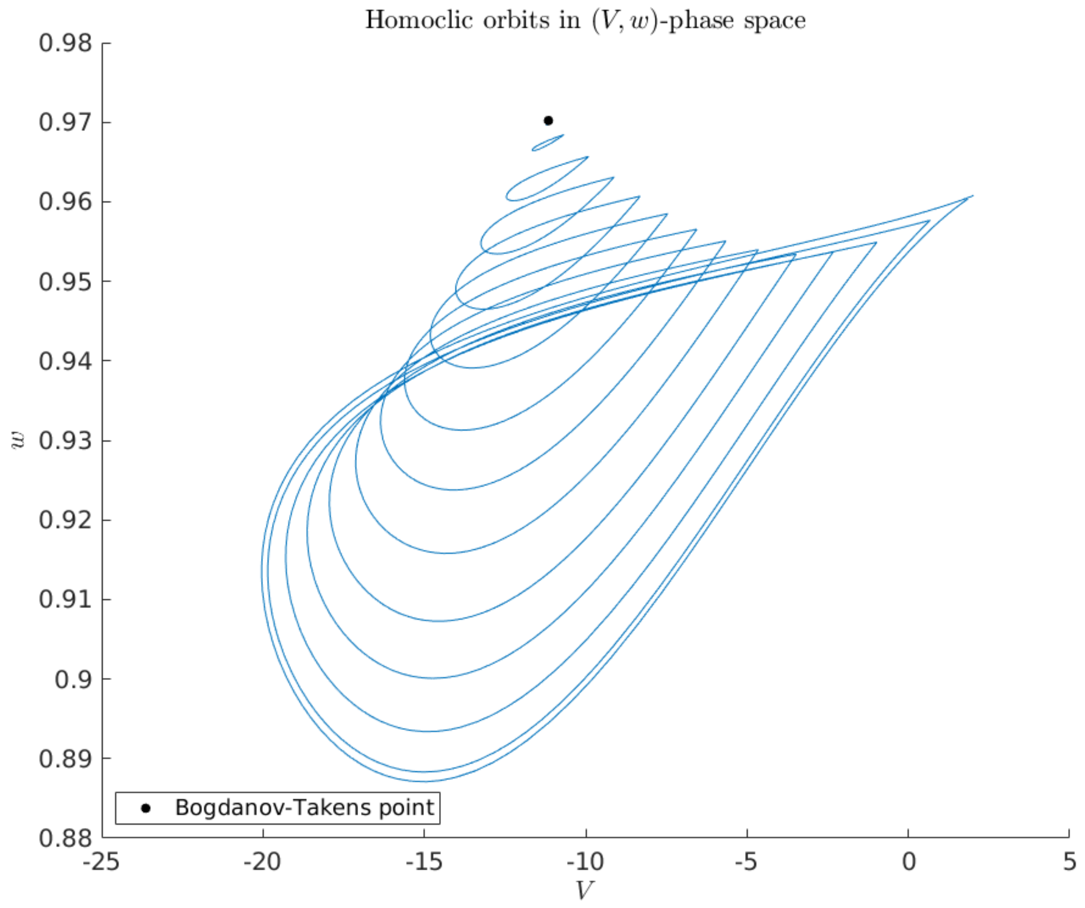
```
hold on
plot(homoclinic_br2(homds.coords(1:homds.nphase:end), 1:5:end), ...
    homoclinic_br2(homds.coords(2:homds.nphase:end), 1:5:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$V$')
ylabel('$w$')
plot(bt2.x(1), bt2.x(2), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthWest')
title('Homoclic orbits in $(V,w)$-phase space')
```



```
bt_index = bt_points_info(3).index;
bt3.x = hopf_br(1:2, bt_index);
bt3.par = p';
bt3.par(ap) = hopf_br(3:4, bt_index);
[hom3_pred, hom3_v_pred] = init_BT_Hom(odefile, bt3, ap, BToptions);
opt.MaxNumPoints  = 200;
homoclinic_br3 = cont(@homoclinic, hom3_pred, hom3_v_pred, opt);
```

```
Center manifold coefficients' accuracy: 1.136868e-13
BT normal form coefficients:
a=7.383698e-04,         b=2.289394e-02
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.0845248
The initial half-return time T: 1284.91
The initial distance eps0: 4.5998e-05
The initial distance eps1: 0.000150094
first point found
tangent vector to first point found
Current step size too small (point 141)
elapsed time  = 5.9 secs
npoints curve = 141
```

```
hold on
plot(homoclinic_br3(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br3(homds.coords(2:homds.nphase:end), 1:10:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$V$')
ylabel('$w$')
plot(bt3.x(1), bt3.x(2), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthWest')
title('Homoclic orbits in $(V,w)$-phase space')
```



```
bt_index = bt_points_info(4).index;
bt4.x = hopf_br(1:2, bt_index);
bt4.par = p';
bt4.par(ap) = hopf_br(3:4, bt_index);
[hom4_pred, hom4_v_pred] = init_BT_Hom(odefile, bt4, ap, BToptions);
homoclinic_br4 = cont(@homoclinic, hom4_pred, hom4_v_pred, opt);
```

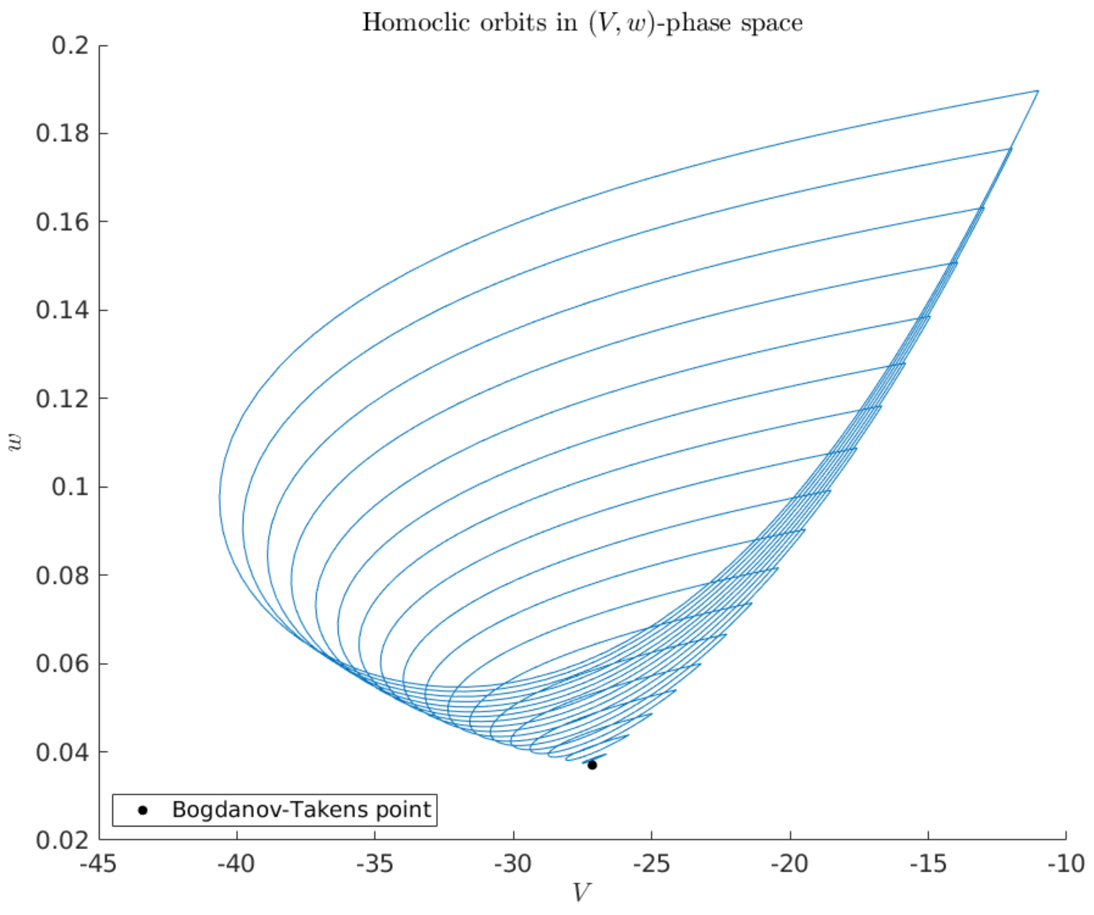
```
Center manifold coefficients' accuracy: 1.136868e-13
BT normal form coefficients:
a=2.766145e-04,          b=1.513072e-02
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.0724949
The initial half-return time T: 2264.92
```

(continues on next page)

(continued from previous page)

```
The initial distance eps0: 3.99117e-05
The initial distance eps1: 0.000129927
first point found
tangent vector to first point found

elapsed time  = 6.4 secs
npoints curve = 200
```

```
hold on
plot(homoclinic_br4(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br4(homds.coords(2:homds.nphase:end), 1:10:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$V$')
ylabel('$w$')
plot(bt4.x(1), bt4.x(2), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthWest')
title('Homoclic orbits in $(V,w)$-phase space')
```



2.16 Continue limit point from the first Bogdanov-Takens point

Next we also continue the limit points emanating from the first and third Bogdanov-Takens points.

```
[lp1_x, lp1_v] = init_BT_LP(odefile, bt1.x, bt1.par, ap);
opt.MaxNumPoints = 2000;
opt.Singularities = 1;
opt.Backward = 1;
[lp1_br, ~, lp1_br_bif] = cont(@limitpoint, lp1_x, lp1_v, opt);
opt.Backward = 0;
opt.MaxNumPoints = 200;
lp1_br_rev = cont(@limitpoint, lp1_x, lp1_v, opt);
```

```
first point found
tangent vector to first point found
label = BT, x = ( 13.611147 0.023136 -227.131888 69.755737 )
(a,b)=(-1.119908e-03, -3.301195e-02)
label = CP, x = ( -10.103668 0.249467 92.105230 6.418208 )
c=5.229002e-05
label = BT, x = ( -27.149904 0.037007 48.225939 21.734143 )
(a,b)=(2.766145e-04, 1.513072e-02)

elapsed time  = 1.3 secs
npoints curve = 2000
first point found
tangent vector to first point found

elapsed time  = 0.2 secs
npoints curve = 200
```

```
[lp2_x, lp2_v] = init_BT_LP(odefile, bt3.x, bt3.par, ap);
opt.MaxNumPoints = 2000;
opt.Singularities = 1;
opt.Backward = 0;
[lp2_br, ~, lp2_br_bif] = cont(@limitpoint, lp2_x, lp2_v, opt);
opt.Backward = 1;
opt.MaxNumPoints = 200;
lp2_br_rev = cont(@limitpoint, lp2_x, lp2_v, opt);
```

```
first point found
tangent vector to first point found
label = BT, x = ( -11.152829 0.970208 519.625365 -63.401902 )
(a,b)=(7.383698e-04, 2.289394e-02)
label = CP, x = ( -2.672919 0.908521 457.907768 -37.108631 )
c=2.750363e-04
label = BT, x = ( 2.815616 0.974408 487.997703 -51.777429 )
(a,b)=(-7.485863e-04, -2.961373e-02)

elapsed time  = 1.1 secs
npoints curve = 2000
first point found
tangent vector to first point found

elapsed time  = 0.1 secs
npoints curve = 200
```

2.17 Extract Cusp points

We see that during the continuation of the limit points two cusp (CP) points were detected in addition to the four known Bogdanov-Takens bifurcation points. We extract the cusp points in the code below.

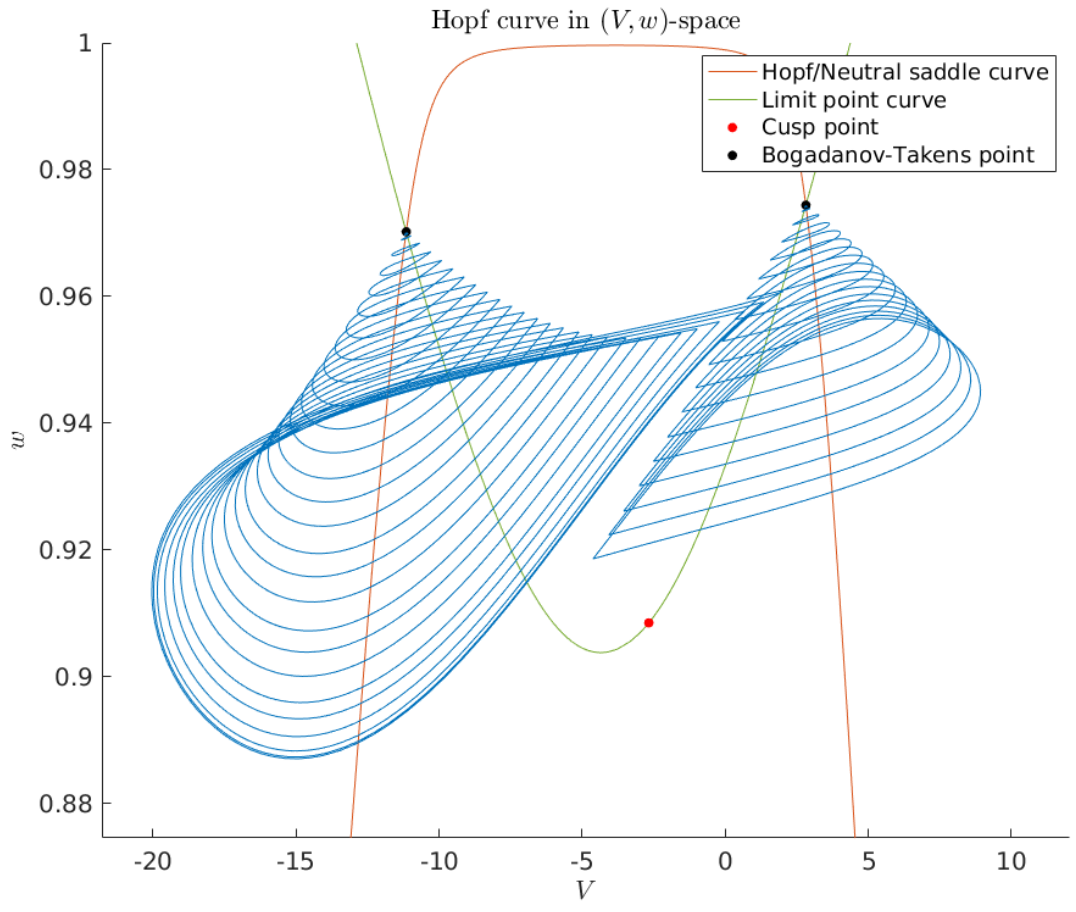
```
cusp1_info = lp1_br_bif(strcmp({lp1_br_bif.label}, 'CP')==1);
cusp2_info = lp2_br_bif(strcmp({lp2_br_bif.label}, 'CP')==1);
Cusp1 = lp1_br(:,cusp1_info.index);
Cusp2 = lp2_br(:,cusp2_info.index);
```

2.18 Plot all curves in (V, w)-phase-space

Here we plot all continued curves and singularities in (V, w)-phase-space. To get a detailed overview we split to plot into two parts.

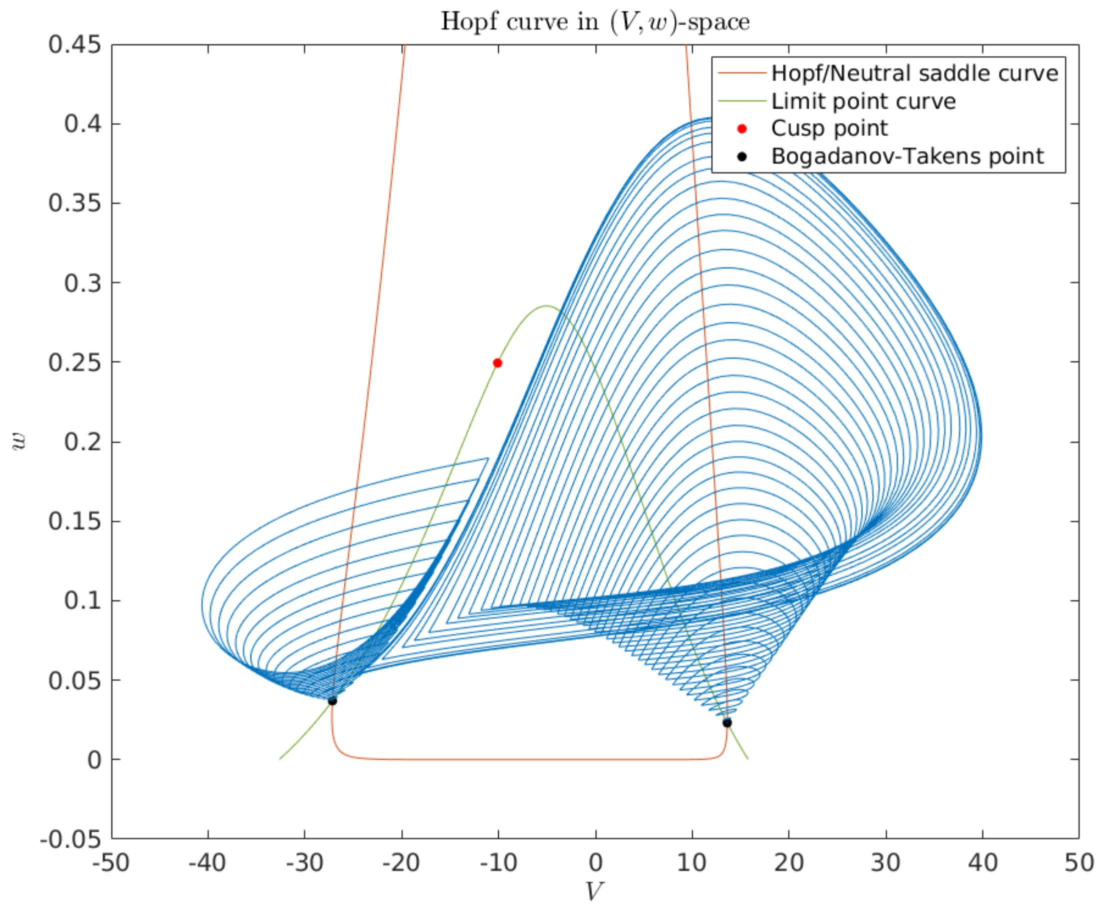
2.18.1 Upper part of the plot

```
%plot --width 1024 --height 800
figure; hold on
homColor  = cm(1,:);
hopfColor = cm(2,:);
foldColor = cm(5,:);
plot(hopf_br(1,:), hopf_br(2,:), 'Color', hopfColor); hold on
plot(lp2_br(1,:), lp2_br(2,:), 'Color', foldColor);
plot(lp2_br_rev(1,:), lp2_br_rev(2,:), 'Color', foldColor, ...
    'HandleVisibility', 'Off');
plot(Cusp2(1), Cusp2(2), '.r' ,'MarkerSize', 20)
plot(BTPoint2(1), BTPoint2(2), '.k' ,'MarkerSize', 20)
plot(BTPoint3(1), BTPoint3(2), '.k' ,'MarkerSize', 20)
plot(homoclinic_br2(homds.coords(1:homds.nphase:end), 1:5:end), ...
     homoclinic_br2(homds.coords(2:homds.nphase:end), 1:5:end), ...
     'Color', homColor, 'DisplayName', 'Off')
plot(homoclinic_br3(homds.coords(1:homds.nphase:end), 1:5:end), ...
     homoclinic_br3(homds.coords(2:homds.nphase:end), 1:5:end), ...
     'Color', homColor, 'HandleVisibility', 'Off')
xlabel('$V$')
ylabel('$w$')
legend({'Hopf/Neutral saddle curve', 'Limit point curve', 'Cusp point', ...
    'Bogadanov-Takens point'}, 'Location', 'NorthEast')
title('Hopf curve in $(V,w)$-space')
axis([-21.7891   12.0474    0.8746    1.0000])
```



2.18.2 Lower part of the plot

```
plot(hopf_br(1,:), hopf_br(2,:), 'Color', hopfColor); hold on
plot(lp1_br(1,:), lp1_br(2,:), 'Color', foldColor);
plot(lp1_br_rev(1,:), lp1_br_rev(2,:), 'Color', foldColor, ...
    'HandleVisibility', 'Off');
plot(Cusp1(1), Cusp1(2), '.r' ,'MarkerSize', 20)
plot(BTPoint1(1), BTPoint1(2), '.k' ,'MarkerSize', 20)
plot(BTPoint4(1), BTPoint4(2), '.k' ,'MarkerSize', 20)
plot(homoclinic_br1(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br1(homds.coords(2:homds.nphase:end), 1:10:end), ...
     'Color', homColor, 'HandleVisibility', 'Off')
plot(homoclinic_br4(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br4(homds.coords(2:homds.nphase:end), 1:10:end), ...
     'Color', homColor, 'HandleVisibility', 'Off')
xlabel('$V$')
ylabel('$w$')
legend({'Hopf/Neutral saddle curve', 'Limit point curve', 'Cusp point', ...
    'Bogadanov-Takens point'}, 'Location', 'NorthEast')
title('Hopf curve in $(V,w)$-space')
axis([-50 50 -0.05 0.45])
```



2.19 Bifurcation plot

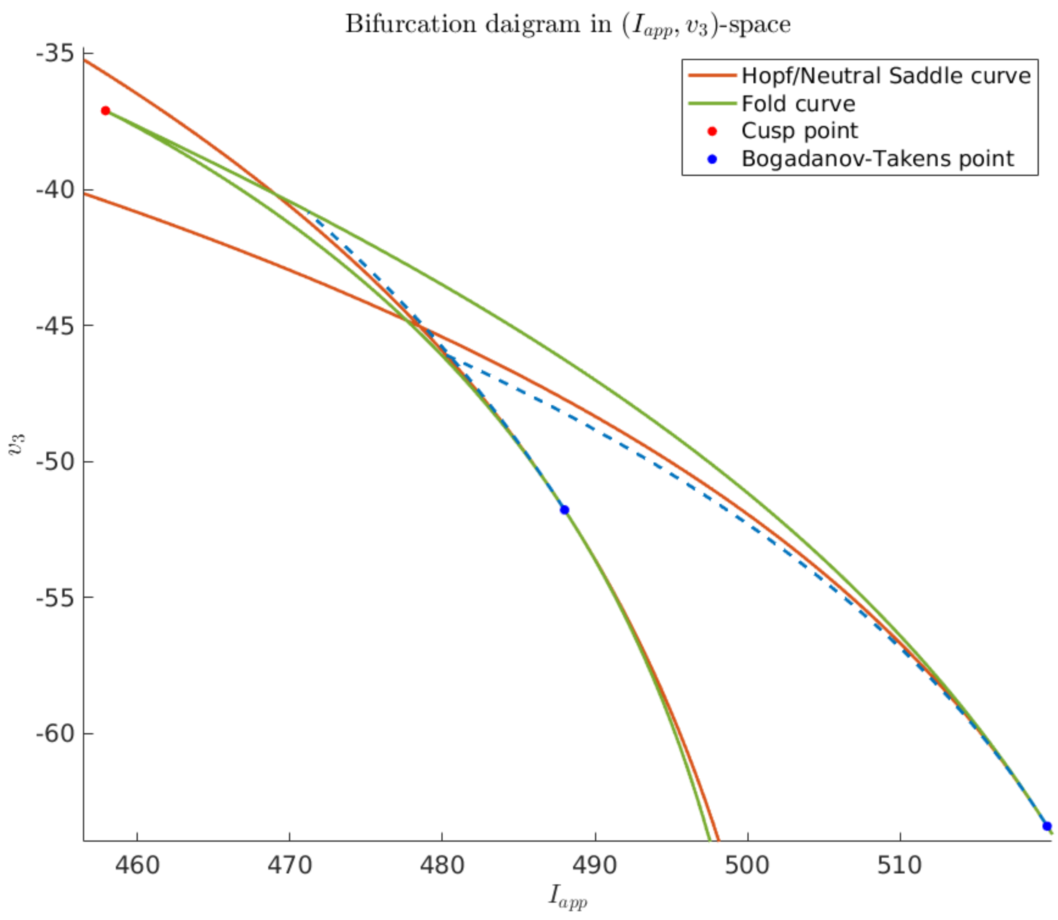
Here we plot all continued curves and singularities in (I_{app}, v_3) parameter space. As before, we split the plot into two parts to obtain a more detailed overview.

```
hold on
plot(hopf_br(3,:), hopf_br(4,:), 'Color', hopfColor, 'linewidth', 2)
plot(lp1_br(3,:), lp1_br(4,:), 'Color', foldColor, 'linewidth', 2)
plot(lp1_br_rev(3,:), lp1_br_rev(4,:), 'Color', foldColor, 'linewidth', 2, ...
    'HandleVisibility', 'Off')
plot(lp2_br(3,:), lp2_br(4,:), 'Color', foldColor, 'linewidth', 2, ...
    'HandleVisibility', 'Off')
plot(lp2_br_rev(3,:), lp2_br_rev(4,:), 'Color', foldColor, 'linewidth', 2, ...
    'HandleVisibility', 'Off')
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
plot(homoclinic_br2(homds.PeriodIdx+1,:), ...
     homoclinic_br2(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
plot(homoclinic_br3(homds.PeriodIdx+1,:), ...
```

(continues on next page)

(continued from previous page)

```
     homoclinic_br3(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
plot(homoclinic_br4(homds.PeriodIdx+1,:), ...
     homoclinic_br4(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
plot(Cusp2(3), Cusp2(4), '.r' ,'MarkerSize', 20)
plot(BTPoint1(3), BTPoint1(4), '.b' ,'MarkerSize', 20)
plot(BTPoint2(3), BTPoint2(4), '.b' ,'MarkerSize', 20)
plot(BTPoint3(3), BTPoint3(4), '.b' ,'MarkerSize', 20)
plot(BTPoint4(3), BTPoint4(4), '.b' ,'MarkerSize', 20)
xlabel('$I_{app}$')
ylabel('$v_3$')
legend({'Hopf/Neutral Saddle curve', 'Fold curve', 'Cusp point',...
    'Bogadanov-Takens point'}, 'Location', 'NorthEast')
title('Bifurcation daigram in $(I_{app},v_3)$-space')
axis([456.5097  519.9544  -63.9705  -34.7564])
```

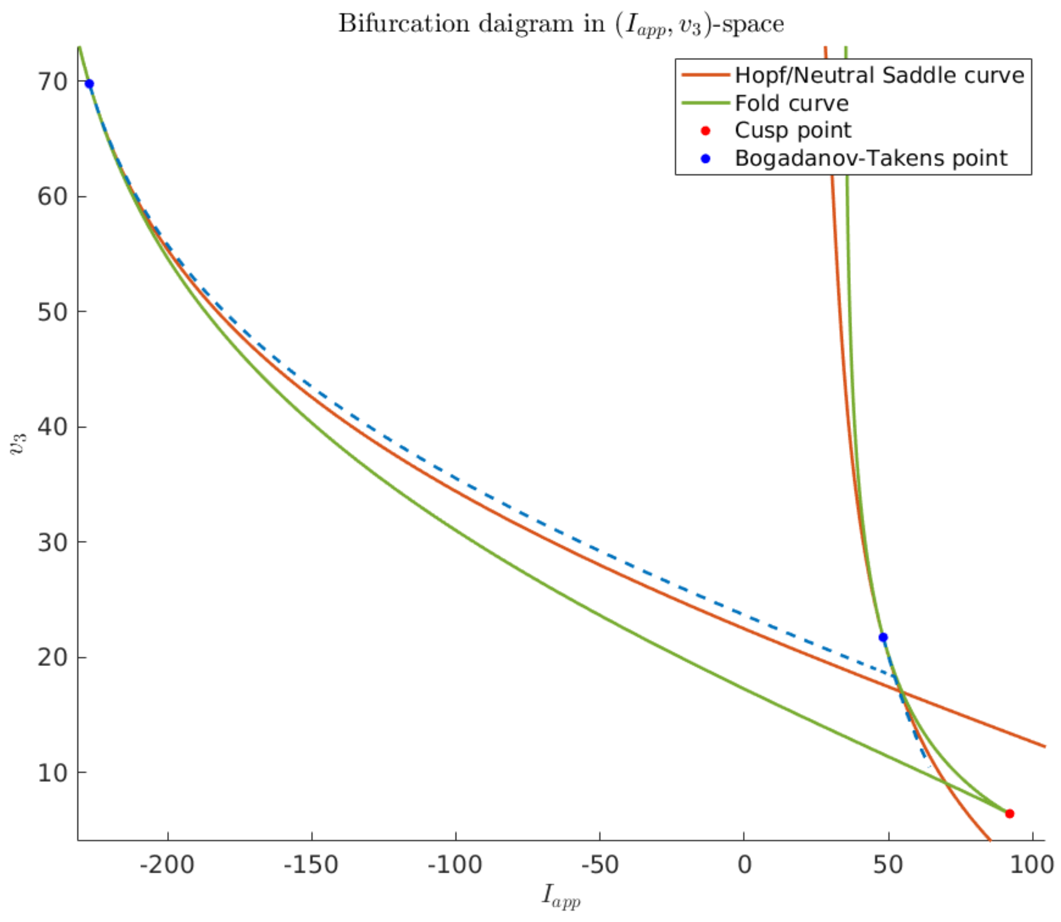


```
hold on
plot(hopf_br(3,:), hopf_br(4,:), 'Color', hopfColor, 'linewidth', 2)
plot(lp1_br(3,:), lp1_br(4,:), 'Color', foldColor, 'linewidth', 2)
plot(lp1_br_rev(3,:), lp1_br_rev(4,:), 'Color', foldColor, 'linewidth', 2, ...
    'HandleVisibility', 'Off')
```

(continues on next page)

(continued from previous page)

```
plot(lp2_br(3,:), lp2_br(4,:), 'Color', foldColor, 'linewidth', 2, ...
    'HandleVisibility', 'Off')
plot(lp2_br_rev(3,:), lp2_br_rev(4,:), 'Color', foldColor, 'linewidth', 2, ...
    'HandleVisibility', 'Off')
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
plot(homoclinic_br2(homds.PeriodIdx+1,:), ...
     homoclinic_br2(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
plot(homoclinic_br3(homds.PeriodIdx+1,:), ...
     homoclinic_br3(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
plot(homoclinic_br4(homds.PeriodIdx+1,:), ...
     homoclinic_br4(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
plot(Cusp1(3), Cusp1(4), '.r' ,'MarkerSize', 20)
plot(BTPoint1(3), BTPoint1(4), '.b' ,'MarkerSize', 20)
plot(BTPoint2(3), BTPoint2(4), '.b' ,'MarkerSize', 20)
plot(BTPoint3(3), BTPoint3(4), '.b' ,'MarkerSize', 20)
plot(BTPoint4(3), BTPoint4(4), '.b' ,'MarkerSize', 20)
xlabel('$I_{app}$')
ylabel('$v_3$')
legend({'Hopf/Neutral Saddle curve', 'Fold curve', 'Cusp point',...
    'Bogadanov-Takens point'}, 'Location', 'NorthEast')
title('Bifurcation daigram in $(I_{app},v_3)$-space')
axis([-231.3697  104.5793    4.0404   73.0235])
```



2.20 Convergence plot

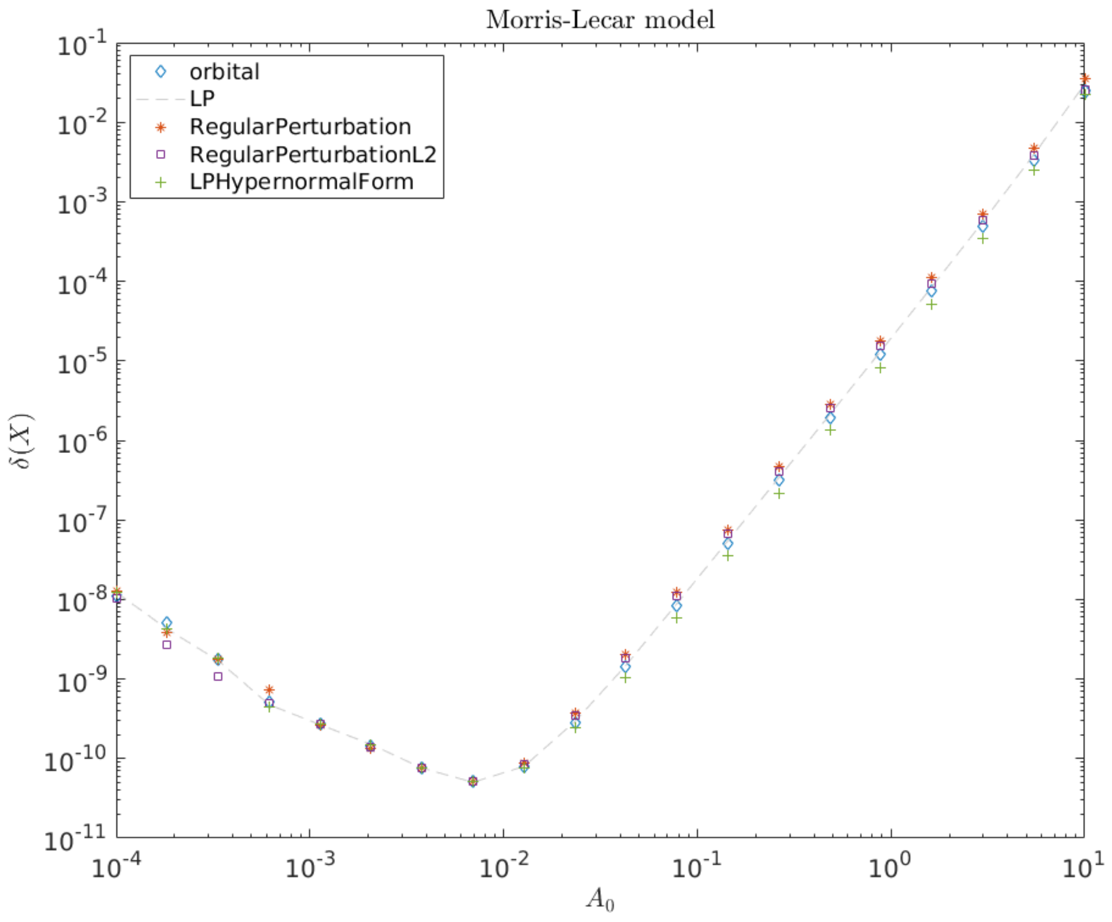
We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 1, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    BToptions.method = methodList{i};
    relativeErrors{i} = zeros(size(amplitudes));
    for j=1:length(amplitudes)
    BToptions.amplitude = amplitudes(j);
```

(continues on next page)

(continued from previous page)

```
    [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, BToptions);
    try
        x_corrected = newtcorr(x_pred, v0);
        relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
    catch
        warning('Did not converge.')
        continue
    end
  end
end

loglog(amplitudes, relativeErrors{1}(:), 'd', ...
       amplitudes, relativeErrors{2}(:), '--', ...
       amplitudes, relativeErrors{3}(:), '*', ...
       amplitudes, relativeErrors{4}(:), 's', ...
       amplitudes, relativeErrors{5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('Morris-Lecar model')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```



Part II

CO-oxidation

CHAPTER

THREE

GENERATE SYSTEM FILES FOR CO-OXIDATION IN A PLATINUM MODEL

In this script the **system files** for the CO-oxidation model

$$\begin{cases} z = 1 - x - y - s, \\ \dot{x} = 2k_1 z^2 - 2k_{-1} x^2 - k_3 xy, \\ \dot{y} = k_2 z - k_{-2} y - k_3 xy, \\ \dot{s} = k_4 (z - \lambda s). \end{cases}$$

are generated. These are used in the *CO-oxidation.ipynb* demo.

3.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

3.2 Set the system name

```
system_name = 'CO_oxidation';
```

3.3 Create coordinates and parameter names as strings

```
coordsnames = {'x', 'y', 's'};
parnames =  {'k1', 'km1', 'k3', 'k2', 'km2', 'k4', 'lambda'};
```

3.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `k1` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});        % create symbol for alpha and delta
par=cell2sym(parnames);   % now alpha1 is par(1) etc
syms(coordsnames{:});     % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

3.5 Define the system

```
z = 1-x-y-s;
dx_dt = 2*k1*z^2 - 2*km1*x^2 - k3*x*y;
dy_dt = k2*z - km2*y - k3*x*y;
ds_dt = k4*(z - lambda*s);
system = [dx_dt; dy_dt; ds_dt];
```

In general there are no modifications needed after this line.

3.6 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

3.7 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

FOUR

CO-OXIDATION IN A PLATINUM MODEL.

In [Khi90], [BIK78], [Bey94] the following chemical model is considered

$$\begin{cases} z = 1 - x - y - s, \\ \dot{x} = 2k_1 z^2 - 2k_{-1} x^2 - k_3 xy, \\ \dot{y} = k_2 z - k_{-2} y - k_3 xy, \\ \dot{s} = k_4 (z - \lambda s). \end{cases}$$

The model describes CO-oxidation in platinum. Here x, y, z and s are the concentrations with $x + y + z + s = 1$ and k_i stand for the reaction rate constants.

4.1 Overview

In this demo we will revisit and extend the results from [BAH15], using the new homoclinic predictor from [Kuz21]. Thus, we will:

- Compute a curve of equilibria, parametrized by k_2.
- Detect two limit points (LP).
- Start continuation from the limit points in two parameters (λ, k_2).
- Detect two Bogdanov-Takens points.
- Start continuation of the homoclinic bifurcation curve emanating from the Bogdanov-Takens points in two parameters (λ, k_2).
- Compare the predicted and computed homoclinic bifurcation curve emanating from the first Bogdanov-Takens point in parameters space.
- Create a convergence plot comparing the different homoclinic approximations derived in [Kuz21].

4.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Equilibrium'])
```

(continues on next page)

(continued from previous page)

```
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
set(groot, 'defaultLineMarkerSize', 20);
```

4.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the script *CO-oxidationGenSym.ipynb*.

```
odefile=@CO_oxidation;
```

4.4 Define equilibrium

We manually define an equilibrium at

$$(x, y, s) = (0.00295, 0.76211, 0.1678), \tag{1}$$

with parameter values $k_1 = 2.5, k_{-1} = 1, k_3 = 10, k_2 = 0, k_{-2} = 0.1, k_4 = 0.0675$ and $\lambda = 1$.

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in *DDE-BifTool*.

```
parnames = {'k1', 'km1', 'k3', 'k2', 'km2', 'k4', 'lambda'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
p(ind.k1)  = 2.5;
p(ind.km1)  = 1;
p(ind.k3)  = 10;
p(ind.km2)  = 0.1;
p(ind.k4)  = 0.0675;
p(ind.lambda)  = 1;
x   = [0.00295; 0.76211; 0.1678];
```

4.5 Continue equilibrium in parameter k_2

To continue the equilibrium (1) in parameter k_2, we first need to obtain a tangent vector to the curve. This is done by the function `init_EP_EP`. Then we use the function `contset` to obain a **struct** containing a list of options which is passed on to the continuer. By adjusting the values of the fields of the `opt` **struct** we set the minimum, initial and maximum step sizes, as well as the direction in which to continue (`opt.Backward`), i.e. in the direction of the obtained tangent vector `v1_pred`, or its negative. We also set the maximum number of points to continue and weather or not to detect bifurcation points (`opt.Singularities`) on the equilibrium curve. For more information about all options available to the *MatCont* continuer and the continuation process in general, we refer to [DGK+08].

Finally, we continue the curve using the function `cont`.

```
[x1_pred, v1_pred] = init_EP_EP(odefile, x, p, ind.k2);
opt = contset;
opt.MinStepsize   = 0.00001;
opt.InitStepsize  = 0.0001;
opt.MaxStepsize   = 0.1;
opt.Backward      = 1;
opt.MaxNumPoints  = 300;
opt.Singularities = 1;
[eqbr_x, ~, eqbr_bif_data] = cont(@equilibrium, x1_pred, v1_pred, opt);
```

```
first point found
tangent vector to first point found
label = LP, x = ( 0.014843 0.696880 0.144138 1.201115 )
a=1.931461e+00
Neutral saddle
label = H , x = ( 0.016833 0.679849 0.151659 1.202834 )
Neutral saddle
label = H , x = ( 0.116822 0.318191 0.282493 1.428482 )
label = LP, x = ( 0.117575 0.316719 0.282853 1.428493 )
a=-2.384097e+01
label = H , x = ( 0.916533 -0.174248 0.128858 -12.529060 )
First Lyapunov coefficient = -2.462095e+00

elapsed time  = 0.3 secs
npoints curve = 300
```

There are two limit points detected (LP), two neutral saddles (H) and one Hopf bifurcation point (H). The **array struct** `eqbr_bif_data` contains information about the detected bifurcation points. We use this to extract the index of the detected bifurcation points on the equilibrium curve `eqbr_x`. The equilibrium curve `eqbr_x` is just a two dimensional array. Each column consists of a point on the curve. The first three rows contain the point (x, y, s) while the last row contains the parameter k_2.

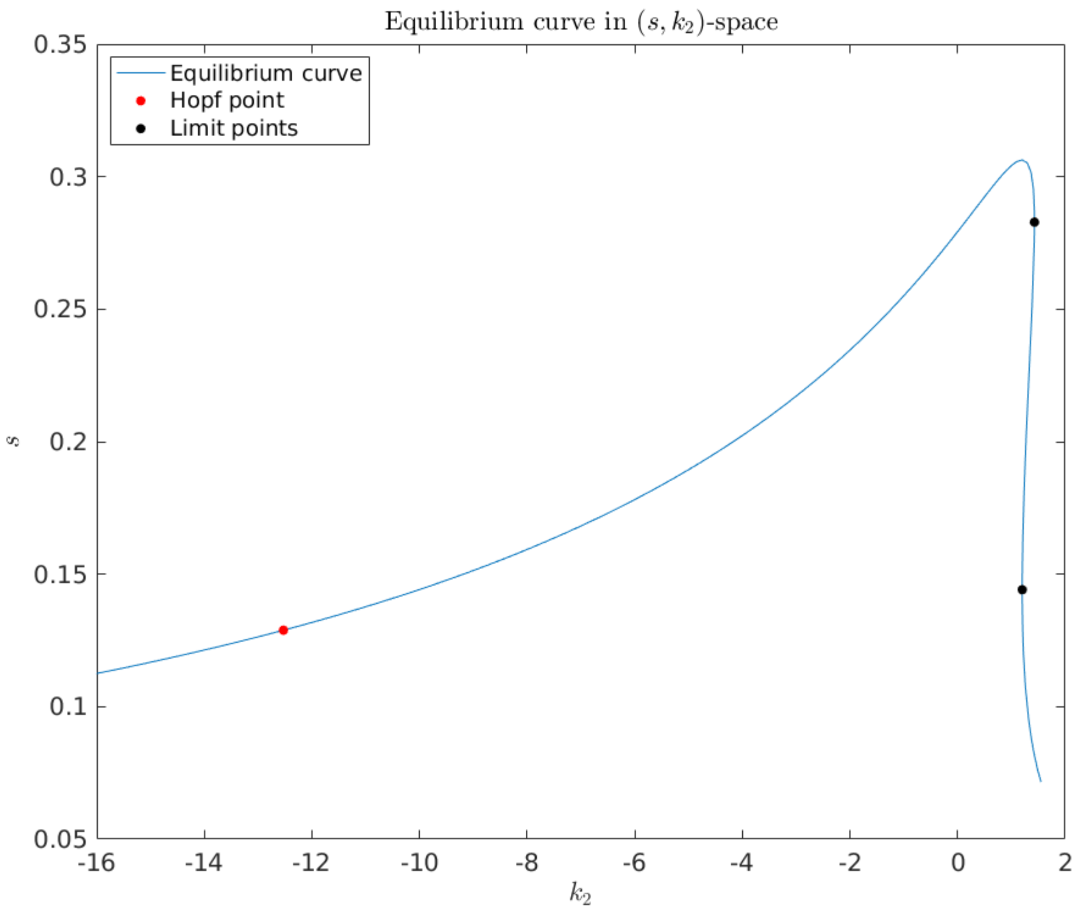
Below we plot the equilibrium curve `eqbr_x`, together with the detected Hopf and limit points, in (s, k_2)-space.

```
%plot --width 1024 --height 800
hopfPointInfo    = eqbr_bif_data(strcmp({eqbr_bif_data.msg}, 'Hopf')==1);
limitPointsInfo = eqbr_bif_data(strcmp({eqbr_bif_data.msg}, 'Limit point')==1);
HopfPoint = eqbr_x(:,hopfPointInfo.index);
limitpoint1 = eqbr_x(:,limitPointsInfo(1).index);
limitpoint2 = eqbr_x(:,limitPointsInfo(2).index);
plot(eqbr_x(4,:), eqbr_x(3,:)); hold on
plot(HopfPoint(4), HopfPoint(3), '.r' ,'MarkerSize', 20)
plot(limitpoint1(4), limitpoint1(3),  '.k' ,'MarkerSize', 20)
plot(limitpoint2(4), limitpoint2(3),  '.k' ,'MarkerSize', 20)
```

(continues on next page)

(continued from previous page)

```
xlabel('$k_2$')
ylabel('$s$')
legend({'Equilibrium curve', 'Hopf point', 'Limit points'}, 'Location', 'NorthWest')
title('Equilibrium curve in $(s,k_2)$-space')
axis([-16 2 0.05 0.35])
```



4.6 Construct limit point

To continue the first limit point in the parameters k_2 and λ we construct a new point `lp` containing the position and parameter values. These are needed to obtain an initial tangent vector - using the function `init_LP_LP` - in the full phase/parameter space. Since, from now on, we will be using the continuation parameter k_2 and λ frequently we assigned these parameters to the variable `ap` (active parameters).

```
ap = [ind.k2 ind.lambda]; % continuation parameters
lp1.par = p;
lp1.par(ap) = eqbr_x(4, limitPointsInfo(1).index);
lp1.x = eqbr_x(1:3, limitPointsInfo(1).index);
[x1, v1] = init_LP_LP(odefile, lp1.x, lp1.par', ap);
```

4.7 Continue limit point in parameters k and T_m

We continue the limit point curve using again the function `cont`. We use the same continuation options as before defined below in the **struct** `opt`, except for the direction of continuation.

```
opt.Backward = 0;
opt.TestTolerance = 1e-12;
opt.MaxTestIters = 10;
[lp_br1, ~, lp_br1_bif] = cont(@limitpoint, x1, v1, opt);
```

```
first point found
tangent vector to first point found
label = BT, x = ( 0.016337 0.638410 0.200456 1.161199 0.722339 )
(a,b)=(-4.822565e-02, -1.937633e+00)
label = CP, x = ( 0.035941 0.352005 0.451370 1.006408 0.355991 )
c=3.627887e-01
label = BT, x = ( 0.115909 0.315467 0.288437 1.417628 0.971398 )
(a,b)=(-8.378444e-02, -2.136280e+00)

elapsed time  = 0.4 secs
npoints curve = 300
```

There are two Bogdanov-Takens bifurcation points (BT) detected on the limit point branch `lp_br1`. The second detected Bogdanov-Takens point coincides with the Bogdanov-Takens point teated in [Bey94].

As with the limit points, information about the detected bifurcation points in stored in the **struct array** `lp_br1_bif`. Below we extract the Bogdanov-Takens bifurcation points.

```
bt_points_info = lp_br1_bif(strcmp({lp_br1_bif.label}, 'BT')==1);
```

4.8 Initial prediction of homoclinic orbit near Bogdanov-Takens point 1 (bt1)

To obtain an initial approximation to the homoclinic solution near the Bogdanov-Takens point we use the function `init_BT_Hom`. Its arguments are the system file (`odefile`), the Bogdanov-Takens point (`bt`) as defined above, the unfolding parameters (`ap`) and an options structure (`BToptions`). The options structure created with the function `BT_Hom_set_options` contains the following fields:

- `ntst` Number of mesh intervals with a default value of 40.
- `ncol` Number of collocation points used in each interval with a default of 4.
- `extravec` Three dimensional boolean row vector indicating which *homoclinic parameters* are selected to be free. The first component refers to the half-return time, while the second and third components refer to the distances from the saddle point to the first, respectively, the last point on the homoclinic orbit. The default value is set to `[0 1 1]`. Thus, the half-return time `T` is fixed.
- `order` The order of the homoclinic approximation used with a default value of 3.
- `amplitude` Desired amplitude of the homoclinic solution. If left empty then a conservative estimate is made, see [Kuz21].
- `TTolerance` Desired distance between the last point on the homoclinic solution and the saddle point. This should at least be smaller than the amplitude. If left empty it is defined by `amplitude*1.0e-03`.

- `HigherOrderTimeReparametrization` Boolean to indicate if a higher order approximation to the non-linear time transformation in the Lindstedt-Poincaré method should be used. This should always be set to `1`. It is only implemented for demonstration purposes.
- `method` Selects the method to be used to approximate the homoclinic solution. The different methods available are:
  - orbital (the default),
  - orbitalv2,
  - LP (Lindstedt-Poincaré with smooth normal form),
  - LPHypernormalForm,
  - RegularPerturbation,
  - RegularPerturbationL2.

  We refer to [Kuz21] for the interpretations.
- `messages` Boolean to indicate if information about selected parameter should be printed the console. The default value is set to `true`.
- `correct` Boolean to indicate if the predicted homoclinic solution should be corrected with Newton. The default value is set to `true`.

Here we will use the default values for the Bogdanov-Takens option structure, except we set the field `correct` to `false` and manually correct the approximation.

```
bt_index = bt_points_info(1).index;
bt1.x = lp_br1(1:3, bt_index);
bt1.par = p';
bt1.par(ap) = lp_br1(4:5, bt_index);
BToptions = BT_Hom_set_options();
BToptions.correct = false;
[x1_pred, v1_pred] = init_BT_Hom(odefile, bt1, ap, BToptions);
```

```
Center manifold coefficients' accuracy: 5.023537e-11
BT normal form coefficients:
a=-4.822565e-02,          b=-1.937633e+00
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.000770702
The initial half-return time T: 1661.8
The initial distance eps0: 4.36004e-07
The initial distance eps1: 1.41437e-06
```

4.9 Correct initial prediction of homoclinic orbit near bt1 with Newton

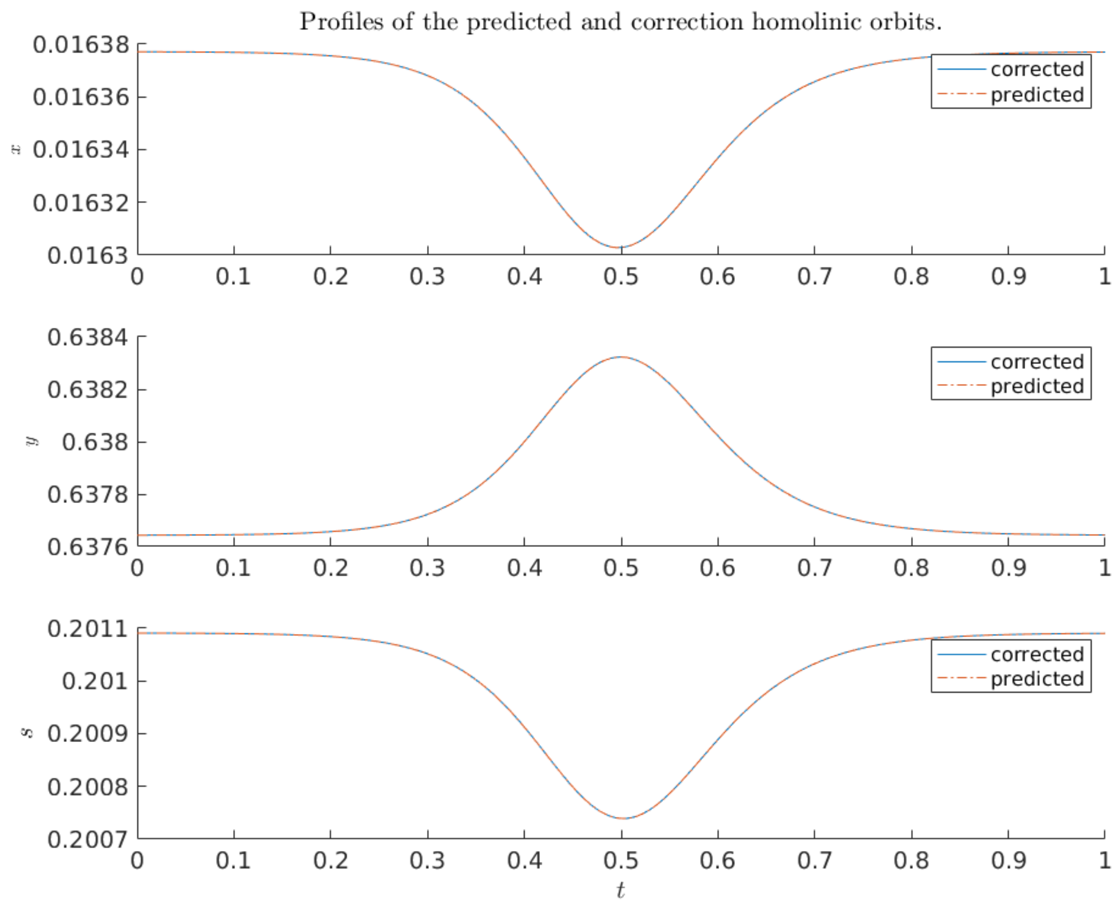
Now that we have an initial prediction for the homoclinic orbit we manually correct it with Newton. After the homoclinic predictor is corrected with the **MatCont** function `newtcorr` we use the function `bt_rearr` (Bogdanov-Takens rearrange) to extract the homoclinic orbit and saddle point from the homoclinic correction.

```
[hom1_x, hom1_v, ~] = newtcorr(x1_pred, v1_pred);
[x1_orbit, x1_saddle] = bt_rearr(hom1_x);
```

4.10 Compare profiles of predicted and corrected solution (bt1)

Using again the **MatCont** function `bt_rearr`, but now on the homoclinic prediction `x1_pred` we compare the profiles of the predicted and corrected homoclinic orbits. We see that they are indistinguishable.

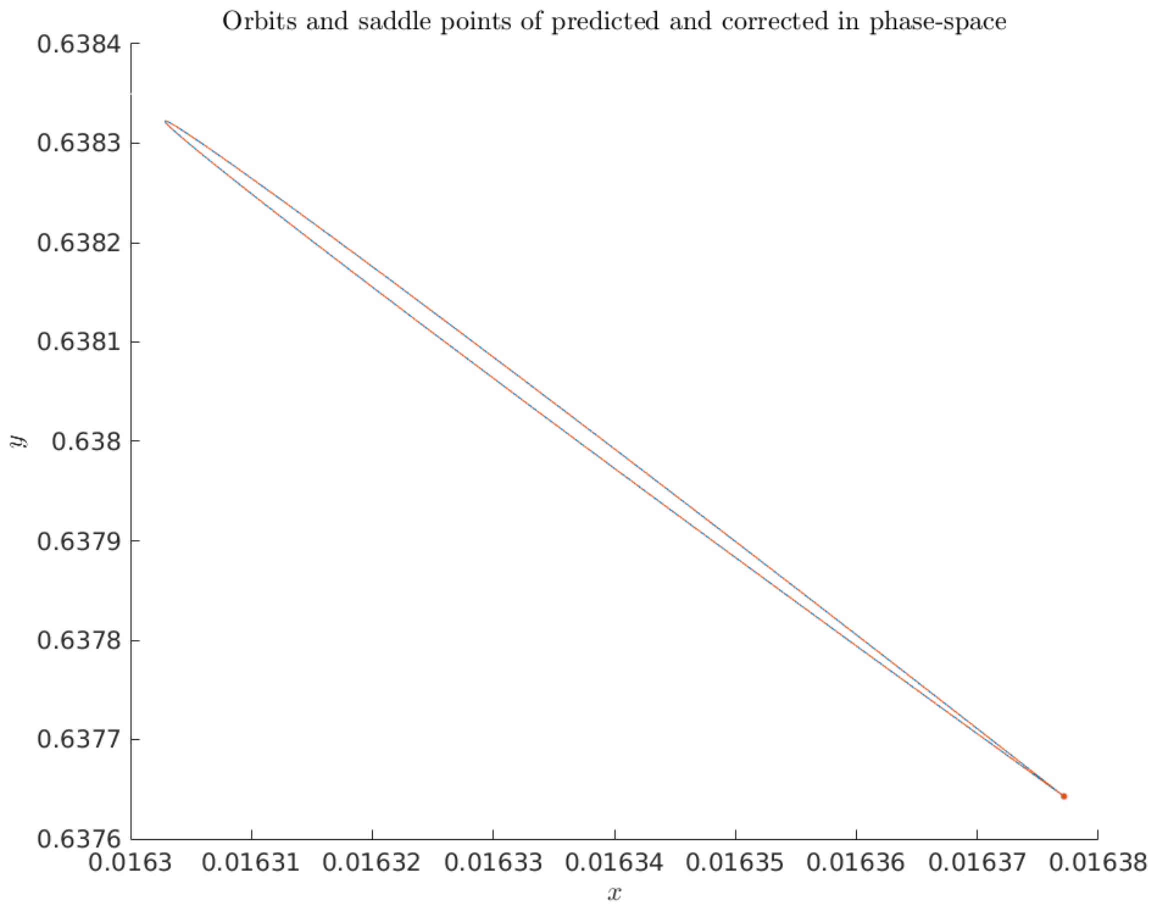
```
global homds
[homoclinic1_pred, saddle1_pred] = bt_rearr(x1_pred);
subplot(3,1,1); hold on;
title('Profiles of the predicted and correction homolinic orbits.')
plot(homds.finemsh, x1_orbit(1:3:end))
plot(homds.finemsh, homoclinic1_pred(1:3:end),'-.')
legend({'corrected', 'predicted'})
ylabel('$x$', 'fontsize', 12,'interpreter', 'latex')
subplot(3,1,2); hold on;
plot(homds.finemsh, x1_orbit(2:3:end))
plot(homds.finemsh, homoclinic1_pred(2:3:end),'-.')
legend({'corrected','predicted'})
ylabel('$y$', 'fontsize', 12,'interpreter', 'latex')
subplot(3,1,3); hold on;
plot(homds.finemsh, x1_orbit(3:3:end))
plot(homds.finemsh, homoclinic1_pred(3:3:end),'-.')
legend({'corrected',  'predicted'})
xlabel('$t$')
ylabel('$s$')
```



4.11 Compare predictor and corrected solution in (x, y) phase-space

Below we compare the predicted and corrected homoclinic orbit in (x, y) phase-space, as well as the predicted and corrected saddle point.

```
hold on
plot(x1_orbit(1:3:end),x1_orbit(2:3:end))
plot(homoclinic1_pred(1:3:end),homoclinic1_pred(2:3:end),'-.')
plot(x1_saddle(1), x1_saddle(2),'.', 'MarkerSize', 12, 'Color', [0 0.4470 0.7410])
plot(saddle1_pred(1), saddle1_pred(2),'.', 'MarkerSize', 12, 'Color', [0.8500, 0.3250,
 ↪ 0.0980])
xlabel('$x$')
ylabel('$y$')
title('Orbits and saddle points of predicted and corrected in phase-space')
```



4.12 Continue homoclinic curve emanating from Bogdanov-Takens point

Having obtain an initial approximation `[hom_x, hom_v]`, where `homo_v` is the tangent vector to the homoclinic curve pointing outwards from the Bogdanov-Takens point, we can start continuation using the function `cont`. Since, we are not interested in bifurcations of the homoclinic orbit, we disable the detection of homoclinic bifurcations `opt.Singularities = 0`. This will reduce the computational cost. We also set the maximum number of continuation steps.

```
opt.Singularities = 0;
opt.MinStepsize  = 1e-06;
opt.InitStepsize = 1e-03;
opt.MaxStepsize  = 1e-02;
opt.MaxNumPoints = 300;
homoclinic_br1 = cont(@homoclinic, hom1_x, hom1_v, opt);
```

```
first point found
tangent vector to first point found

elapsed time  = 13.6 secs
```

(continues on next page)

(continued from previous page)

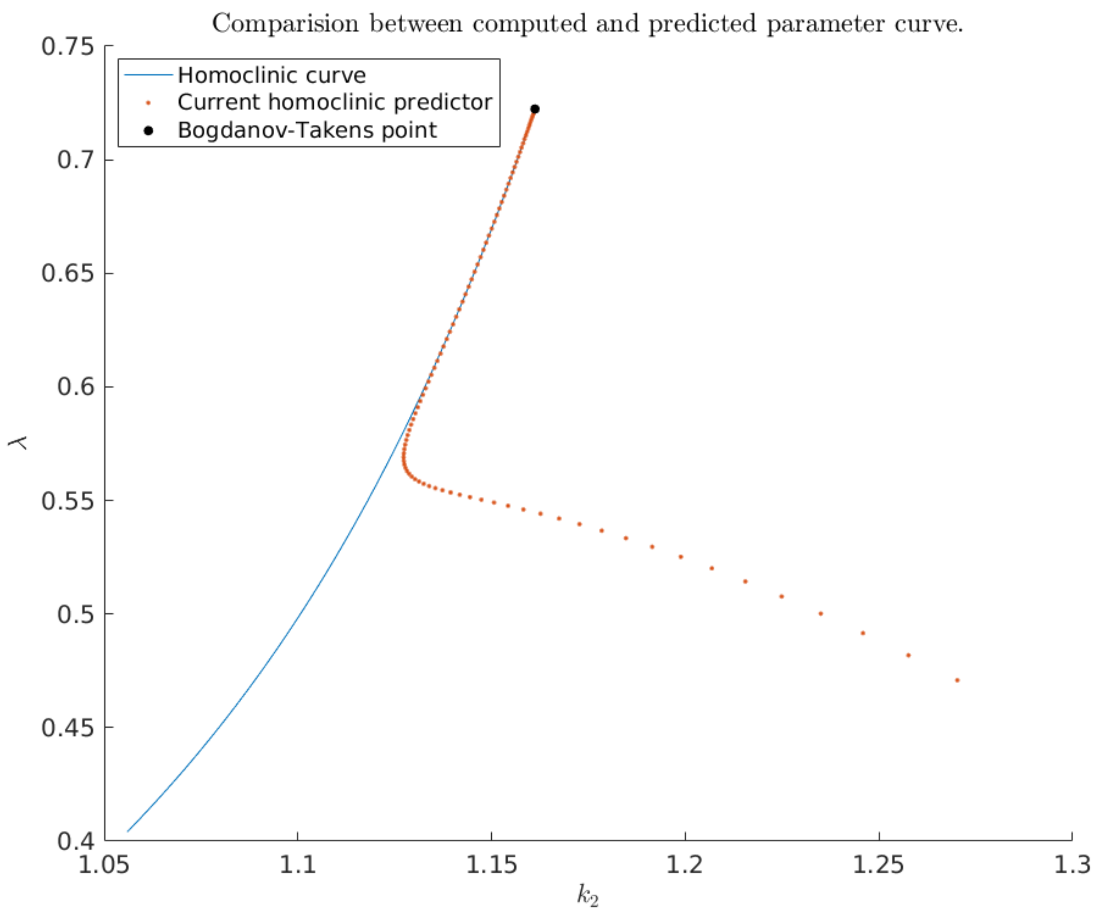
```
npoints curve = 300
```

4.13 Compare predicted with computed parameters emanating from bt1

Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm` to obtain the parameter-dependent normal form coefficients and the coefficients between the parameters of the system and the parameters on the center manifold, see [Kuz21].

```
hold on
% plot computed parameter curve
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent normal form coefficients
bt1 = BT_nmfm_orbital(odefile, bt1, ap, BToptions);
a   = bt1.nmfm.a;
b   = bt1.nmfm.b;
K10 = bt1.nmfm.K10;
K01 = bt1.nmfm.K01;
K02 = bt1.nmfm.K02;
K11 = bt1.nmfm.K11;
K03 = bt1.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 1.8);
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alpha = bt1.par(ap) + alpha;
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.')
% plot Bogdanov-Takens point
plot(bt1.par(ind.k2), bt1.par(ind.lambda), '.k', 'MarkerSize', 20)
% set axis labels and legend
xlabel('$k_2$')
ylabel('$\lambda$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'NorthWest')
title('Comparision between computed and predicted parameter curve.')
```

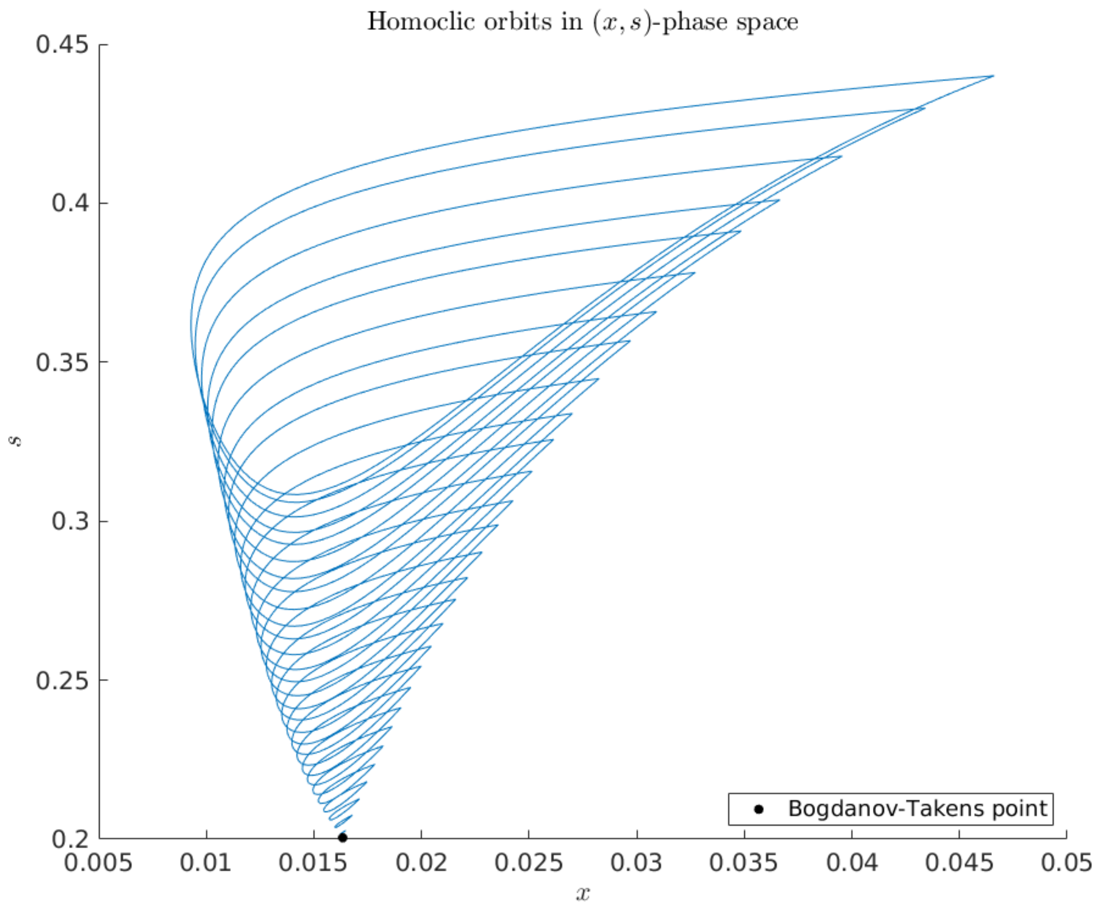
```
Center manifold coefficients' accuracy: 5.023537e-11
```



4.14 Bifurcation diagram in (x, s) phase-space

To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (x, s) phase-space.

```
hold on
%plot naive
plot(homoclinic_br1(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br1(homds.coords(3:homds.nphase:end), 1:10:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$x$')
ylabel('$s$')
plot(bt1.x(1), bt1.x(3), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(x,s)$-phase space')
```



4.15 Continue homoclinic curve emanating from Bogdanov-Takens point

We continue the homoclinic orbit as usual with the **MatCont** function `cont`. However, since we already corrected the initial prediction to the homoclinic orbits, we use this correction as an initial starting point of continuation in stead of the predictor. If one isn't interested in the correction of the initial point, the prediction could be used directly as argument to `cont`. Lastly, since, here, we are not interested in bifurcations of the homoclinic orbit, we disable the detection of homoclinic bifurcations. This will reduce the computational cost.

```
bt2_index = bt_points_info(2).index;
bt2.x = lp_br1(1:3, bt2_index);
bt2.par = p';
bt2.par(ap) = lp_br1(4:5, bt2_index);
options = BT_Hom_set_options();
[x2_pred, v2_pred] = init_BT_Hom(odefile, bt2,  ap, options);
homoclinic_br2 = cont(@homoclinic, x2_pred, v2_pred, opt);
```

```
Center manifold coefficients' accuracy: 2.761569e-11
BT normal form coefficients:
a=-8.378444e-02,         b=-2.136280e+00
```

(continues on next page)

(continued from previous page)

```
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.00110154
The initial half-return time T: 1052.82
The initial distance eps0: 6.38083e-07
The initial distance eps1: 2.06064e-06
first point found
tangent vector to first point found

elapsed time  = 11.2 secs
npoints curve = 300
```

4.16 Continue Hopf point from the first Bogdanov-Takens point

We convert the first Bogdanov-Takens point into a Hopf point in order to start continuation of the Hopf bifurcation curve.

```
[hopf_x, hopf_v] = init_BT_H(odefile, bt1.x, bt1.par, ap);
```

We continue the Hopf point using the function `cont`. Then we try to filter out the neutral saddle points by inspecting the eigenvalues.

```
opt.MaxNumPoints = 2000;
opt.Singularities = 1;
opt.Eigenvalues = 1;
[hopf_br, ~, hopf_br_bif, ~, eigenvalues] = cont(@hopf, hopf_x, hopf_v, opt);
neutral_saddle_br = hopf_br(:,abs(imag(eigenvalues(2,:))) < 0.00001);
hopf_br_corrected = hopf_br(:,abs(imag(eigenvalues(2,:))) >= 0.00001);
```

```
first point found
tangent vector to first point found
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
 ↪RCOND =  2.031561e-16.
> In nf_H (line 27)
In hopf>testf (line 150)
In cont>EvalTestFunc (line 810)
In cont (line 506)

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
 ↪RCOND =  6.039544e-17.
> In nf_H (line 28)
In hopf>testf (line 150)
In cont>EvalTestFunc (line 810)
In cont (line 506)

label = BT, x = ( 0.016337 0.638410 0.200456 1.161199 0.722339 0.000000 )
(a,b)=(-4.822565e-02, -1.937633e+00)
label = BT, x = ( 0.115909 0.315467 0.288437 1.417628 0.971398 0.000000 )
(a,b)=(8.378444e-02, 2.136280e+00)
label = GH, x = ( 0.064311 0.211095 0.554870 0.924255 0.305879 0.003512 )
l2=-2.401234e+02
label = GH, x = ( 0.018022 0.368238 0.497968 0.891319 0.232487 0.003324 )
l2=-7.769003e+02
Closed curve detected at step 1241
```

(continues on next page)

(continued from previous page)

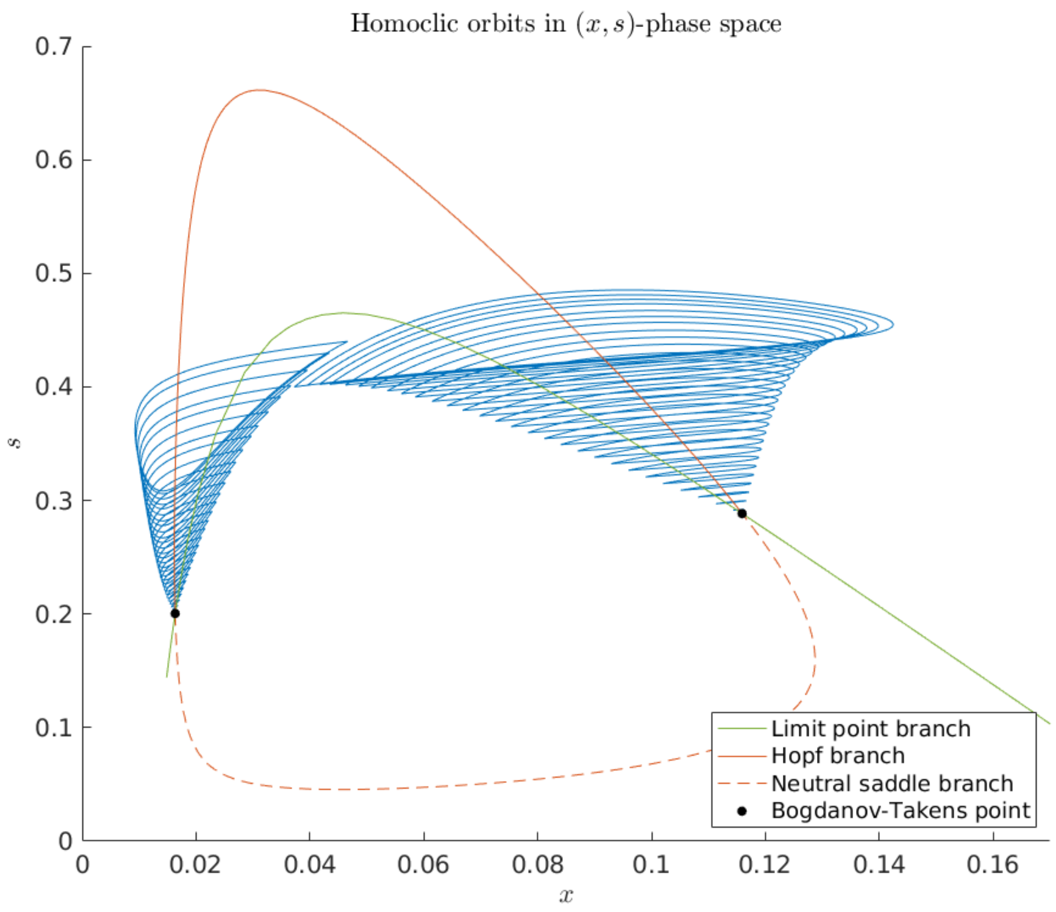
```
elapsed time  = 1.3 secs
npoints curve = 1241
```

We see that in addition to the two Bogdanov-Takens points already detected before, there are also two generalized Hopf points (GH) detected.

4.17 Bifurcation diagram in phase-space (x, s)

Below we plot the computed homoclinic orbits in (x, s) phase-space.

```
hold on
colormap = lines();
homColor  = colormap(1,:);
hopfColor = colormap(2,:);
foldColor = colormap(5,:);
plot(homoclinic_br1(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br1(homds.coords(3:homds.nphase:end), 1:10:end), ...
     'Color', homColor, 'HandleVisibility', 'Off')
plot(homoclinic_br2(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br2(homds.coords(3:homds.nphase:end), 1:10:end), ...
     'Color', homColor, 'HandleVisibility', 'Off')
plot(lp_br1(1,:), lp_br1(3,:), 'Color', foldColor)
plot(hopf_br_corrected(1,:), hopf_br_corrected(3,:), 'Color', hopfColor)
plot(neutral_saddle_br(1,1:end-1), neutral_saddle_br(3,1:end-1), '--', ...
    'Color', hopfColor)
xlabel('$x$')
ylabel('$s$')
plot(bt1.x(1), bt1.x(3), '.k')
plot(bt2.x(1), bt2.x(3), '.k', 'HandleVisibility', 'Off')
legend('Limit point branch', 'Hopf branch', 'Neutral saddle branch', ...
    'Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(x,s)$-phase space')
axis([0 0.17 0 0.7])
```



4.18 Bifurcation diagram in parameter-space (k_2, λ)

We finish this notebook by plotting the computed homoclinic orbits in (x, s) phase-space.

Below we extract the cusp bifurcation point and the generalized Hopf points on the limit point branch and Hopf branch respectively.

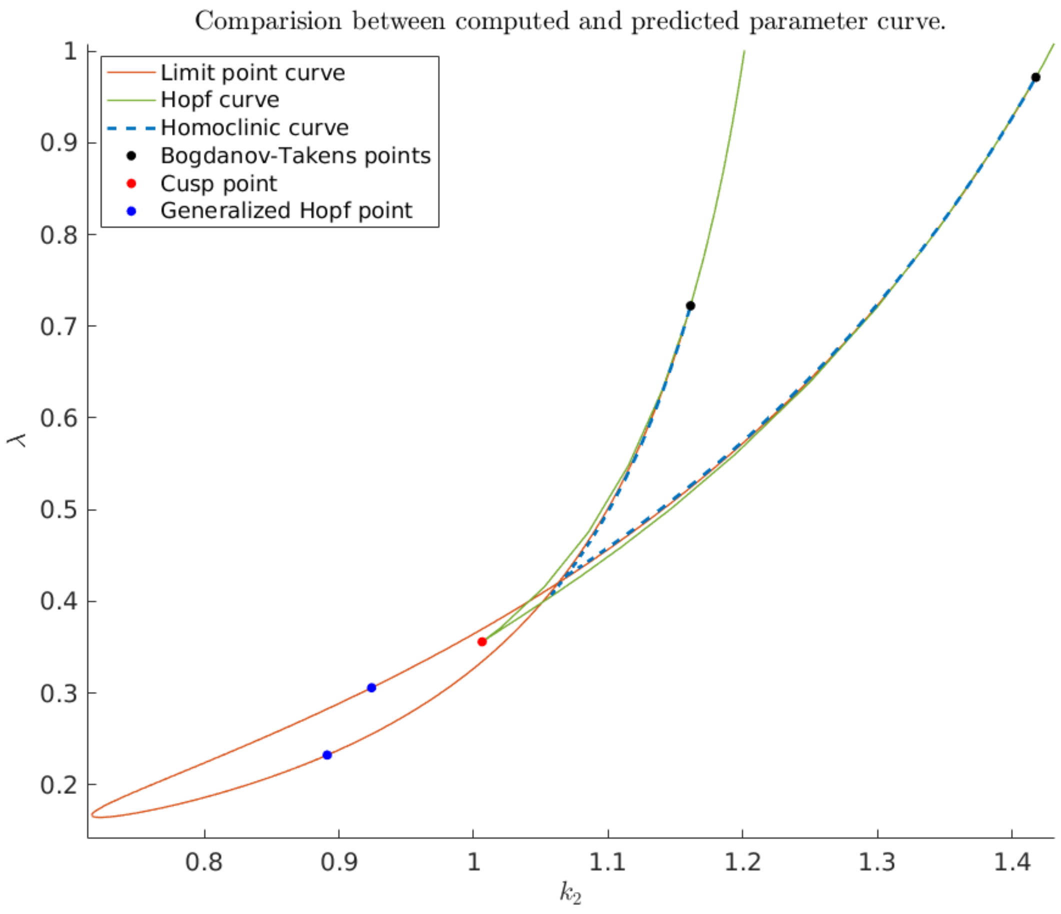
```
cusp_point_info = lp_br1_bif(strcmp({lp_br1_bif.label}, 'CP')==1);
genh_info = hopf_br_bif(strcmp({hopf_br_bif.label}, 'GH')==1);
cusp = lp_br1(:,cusp_point_info.index);
genh1 = hopf_br(:,genh_info(1).index);
genh2 = hopf_br(:,genh_info(2).index);
```

```
hold on
% plot computed parameter curve
plot(hopf_br_corrected(4,:), hopf_br_corrected(5,:), 'Color', hopfColor, 'linewidth',␣
↪1)
plot(lp_br1(4,:), lp_br1(5,:), 'Color', foldColor, 'linewidth', 1)
% plot homoclinic curves
plot(homoclinic_br1(homds.PeriodIdx+1,:), homoclinic_br1(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2)
```

(continues on next page)

(continued from previous page)

```
plot(homoclinic_br2(homds.PeriodIdx+1,:), homoclinic_br2(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'HandleVisibility', 'Off', 'linewidth', 2)
% plot Bogdanov-Takens point
plot(bt1.par(ind.k2), bt1.par(ind.lambda), '.k')
plot(bt2.par(ind.k2), bt2.par(ind.lambda), '.k', 'HandleVisibility', 'Off')
plot(cusp(4), cusp(5), '.r')
plot(genh1(4), genh1(5), '.b')
plot(genh2(4), genh2(5), '.b', 'HandleVisibility', 'Off')
% set axis, labels and legend
xlabel('$k_2$')
ylabel('$\lambda$')
legend({'Limit point curve', 'Hopf curve', 'Homoclinic curve', ...
    'Bogdanov-Takens points', 'Cusp point', 'Generalized Hopf point'}, ...
    'Location', 'NorthWest')
title('Comparision between computed and predicted parameter curve.')
axis([0.7130    1.4320    0.1413    1.0081])
```



4.19 Convergence plot

We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

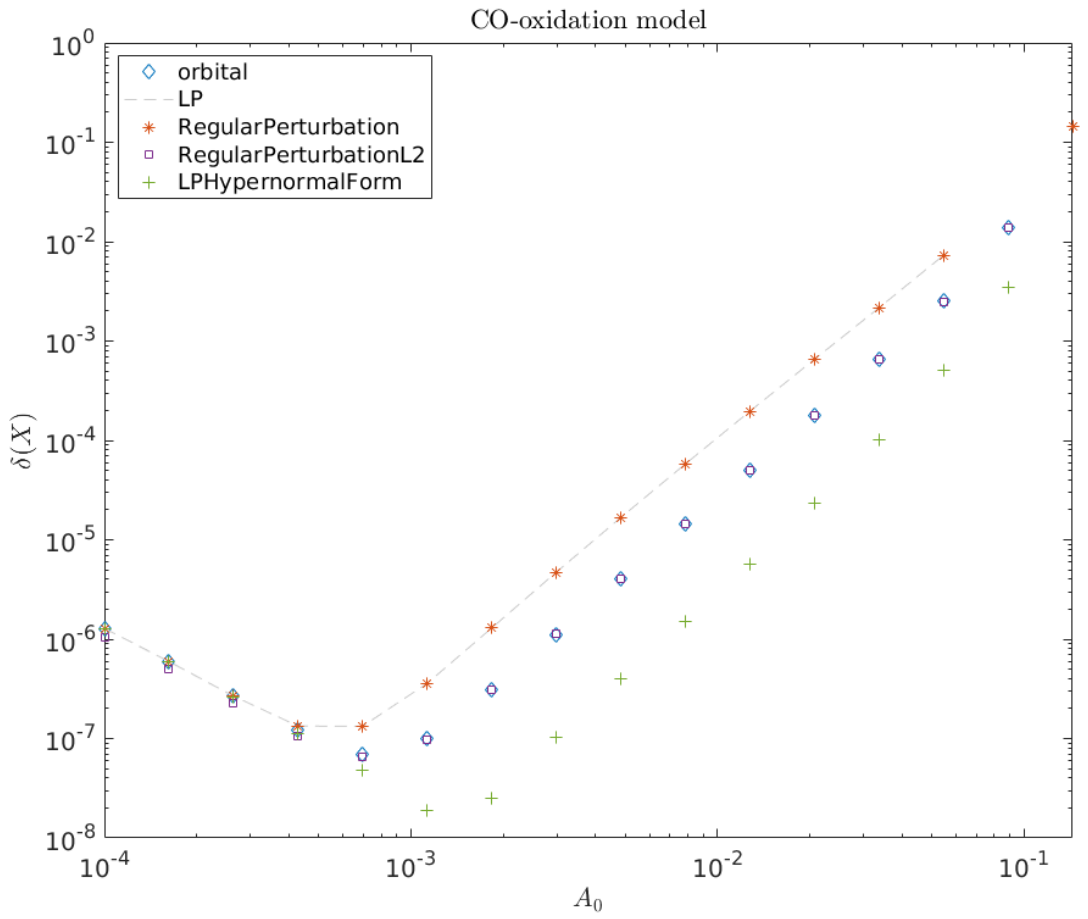
```
set(groot, 'defaultLineMarkerSize', 7);
options = BT_Hom_set_options();
options.TTolerance = 1e-05;
options.messages = false;
options.correct = false;

amplitudes = logspace(-4, 0, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    options.method = methodList{i};
    relativeErrors{i} = zeros(size(amplitudes));
    for j=1:length(amplitudes)
    options.amplitude = amplitudes(j);
    [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, options);
    try
        x_corrected = newtcorr(x_pred, v0);
        relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
    catch
        warning('Did not converge.')
        continue
    end
  end
end

cm = lines();
loglog(amplitudes, relativeErrors{1}(:), 'd', ...
       amplitudes, relativeErrors{2}(:), '--', ...
       amplitudes, relativeErrors{3}(:), '*', ...
       amplitudes, relativeErrors{4}(:), 's', ...
       amplitudes, relativeErrors{5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('CO-oxidation model')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```

```
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
```

(continues on next page)

(continued from previous page)

```
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
```



Part III

Predator Prey

CHAPTER

FIVE

GENERATE SYSTEM FILES FOR PREDATOR-PREY SYSTEM WITH CONSTANT RATE HARVESTING

In this script the **system files** for the predator-prey system given by

$$\begin{cases} \dot{x} = rx\left(1 - \frac{x}{k}\right) - y\frac{x}{e+x}, \\ \dot{y} = y\left(-d + \frac{x}{e+x}\right) - h, \end{cases}$$

are generated. These are used in the *predator prey* demo.

5.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

5.2 Set the system name

```
system_name = 'PredatorPrey';
```

5.3 Create coordinates and parameter names as strings

```
coordsnames = {'x', 'y'};
parnames = {'d', 'h'};
```

5.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `alpha` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});        % create symbol for alpha and delta
par=cell2sym(parnames);   % now alpha1 is par(1) etc
syms(coordsnames{:});     % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

5.5 Define fixed parameters

```
r = 1;
e = 1;
k = 2;
```

5.6 Define the system

```
dx_dt = r*x*(1-x/k) - y*x/(e+x);
dy_dt = y*(-d+x/(e+x)) - h;
system = [dx_dt; dy_dt];
```

In general there are no modifications needed after this line.

5.7 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

5.8 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

SIX

PREDATOR-PREY SYSTEM WITH CONSTANT HARVESTING RATE

In [BS79] the following predator-prey system with constant harvesting rate is considered

$$\begin{cases} \dot{x} = rx\left(1 - \dfrac{x}{k}\right) - y\dfrac{x}{e+x}, \\ \dot{y} = y\left(-d + \dfrac{x}{e+x}\right) - h. \end{cases} \tag{1}$$

Here the parameters have the following interpretations:

- k is the carrying capacity of the prey population,
- d is the death rate of the predator,
- r is the intrinsic growth rate of the prey population,
- and h is the harvesting rate.

The function $x \mapsto \frac{x}{e+x}$ is often called the functional response of Holling type II. The authors in [XR99] show the existence of a Bogdanov-Takens bifurcation in (1) and sketch the global bifurcation diagram including the homoclinic curve which emanates from the Bogdanov-Takens point. We study the occurrence of homoclinic orbits that emanate from the computed Bogdanov-Takens point using *MatCont*.

6.1 Overview

In this demo we will

- Define an analytically derived Bogdanov-Takens point.
- Start continuation of the homoclinic branch emanating from the Bogdanov-Takens point in two parameters (d, h) using the new homoclinic smooth orbital predictor from [Kuz21].
- Compare the predicted and computed homoclinic bifurcation curve emanating from the Bogdanov-Takens point in parameters space.
- Compare a range of predictors for the homoclinic solutions emanating from the Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space.
- Create a convergence plot of the different homoclinic approximations derived in [Kuz21].

6.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Equilibrium'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitPointCycle'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
```

6.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *PredatorPreyGenSym.ipynb*.

```
odefile=@PredatorPrey;
```

6.4 Define Bogdanov-Takens point manually

There are two equilibrium points in (1). One is given by

$$\begin{cases} x_1 &= \frac{\sqrt{9d^2+8dh-12d-8h+4}-d+2}{2-2d}, \\ x_2 &= -\frac{\sqrt{9d^2+8dh-12d-8h+4}+d(4h+3)-4h-2}{4(d-1)^2}. \end{cases} \tag{2}$$

By evaluating the Jacobian of (1)

$$J = \begin{pmatrix} \frac{xy}{(x+1)^2} - \frac{y}{x+1} - x + 1 & -\frac{x}{x+1} \\ \left(\frac{1}{x+1} - \frac{x}{(x+1)^2}\right) y & \frac{x}{x+1} - d \end{pmatrix}$$

at the equilibrium (2) and subsequently solving for a double zero of the characteristic equation of J for (d, h) yields

$$d = \frac{1}{9}\left(8 - \frac{11}{\sqrt[3]{548-18\sqrt{894}}} - \frac{\sqrt[3]{274-9\sqrt{894}}}{2^{2/3}}\right),$$

$$h = \frac{93\ 2^{2/3} - 9\sqrt[6]{2}\sqrt{447} - 975\sqrt[3]{\frac{2}{274-9\sqrt{894}}} + \frac{36\ 2^{5/6}\sqrt{447}}{\sqrt[3]{274-9\sqrt{894}}} + 30\sqrt[3]{274-9\sqrt{894}}}{88\ 2^{2/3} + 16\sqrt[3]{274-9\sqrt{894}} + 8\sqrt[3]{2}\left(274-9\sqrt{894}\right)^{2/3}}.$$

```
d = (1/9).*(8+(-11).*(548+(-18).*894.^(1/2)).^(-1/3)+(-1).*2.^(-2/3).* ...
(274+(-9).*894.^(1/2)).^(1/3));
h = (93.*2.^(2/3)+(-9).*2.^(1/6).*447.^(1/2)+(-975).*(2.*(274+(-9).* ...
    894.^(1/2)).^(-1)).^(1/3)+36.*2.^(5/6).*447.^(1/2).*(274+(-9).* ...
    894.^(1/2)).^(-1/3)+30.*(274+(-9).*894.^(1/2)).^(1/3)).*(88.*2.^( ...
    2/3)+16.*(274+(-9).*894.^(1/2)).^(1/3)+8.*2.^(1/3).*(274+(-9).* ...
    894.^(1/2)).^(2/3)).^(-1);
x = (1/2).*((-2)+d).*((-1)+d).^(-1);
y = (1/8).*((-1)+d).^(-2).*(8+(-18).*d+9.*d.^2);
bt.x = [x; y];
bt.par = [d; h];
```

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in *DDE-BifTool*.

```
parnames = {'d', 'h'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
```

6.5 Continue homoclinic curve emanating from the Bogdanov-Takens point

To continue the homoclinic curve emanating from the Bogdanov-Takens point we use the functions `BT_Hom_set_options` and `init_BT_Hom` to obtain an initial approximation to the homoclinic solution (`hom_x`) as well as a tangent vector to the discretized homoclinic solution (`hom_v`) as described in *Initial prediction of homoclinic orbit near Bogdanov-Takens point 1*.

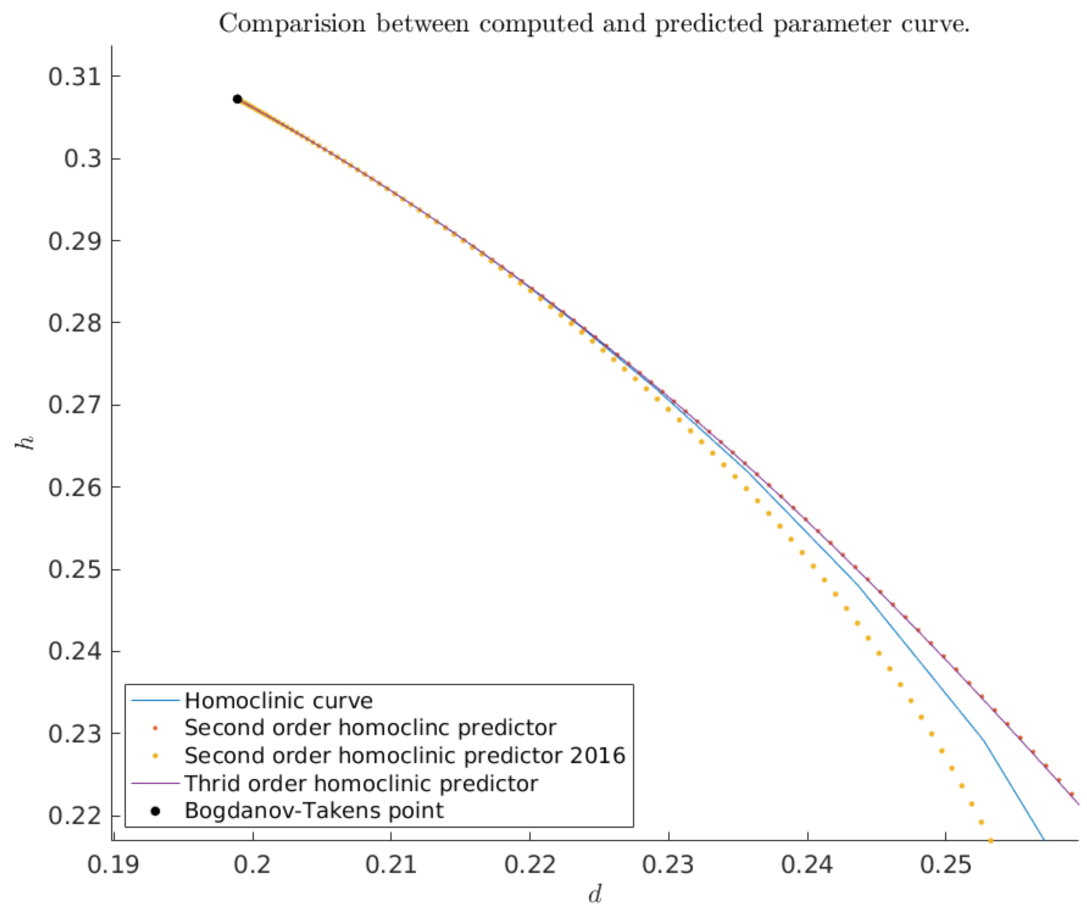
```
ap = [ind.d, ind.h];
BToptions = BT_Hom_set_options();
[hom_x, hom_v] = init_BT_Hom(odefile, bt,  ap, BToptions);
opt = contset;
opt.MaxStepsize = 5;
opt.Singularities = 0;
opt.MaxNumPoints = 200;
homoclinic_br = cont(@homoclinic, hom_x, hom_v, opt);
```

```
Center manifold coefficients' accuracy: 9.992007e-16
BT normal form coefficients:
a=1.798257e-01,         b=-3.782866e-01
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.0753982
The initial half-return time T: 86.8246
The initial distance eps0: 0.000144111
The initial distance eps1: 4.47679e-05
first point found
tangent vector to first point found
Current step size too small (point 118)
elapsed time  = 6.3 secs
npoints curve = 118
```

6.6 Compare predicted with computed parameters

Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm_orbital` to obtain the smooth orbital normal form coefficients, i.e. a and b, and the coefficients of the transformation K between the parameters of the system and the parameters of the smooth orbital normal form on the center manifold, see [Kuz21]. By using the transformation K as given in [AHGKM16] where K_{11} is not included for the second order approximation we see that including K_{11} as done in [Kuz21] does indeed lead to a better approximation.

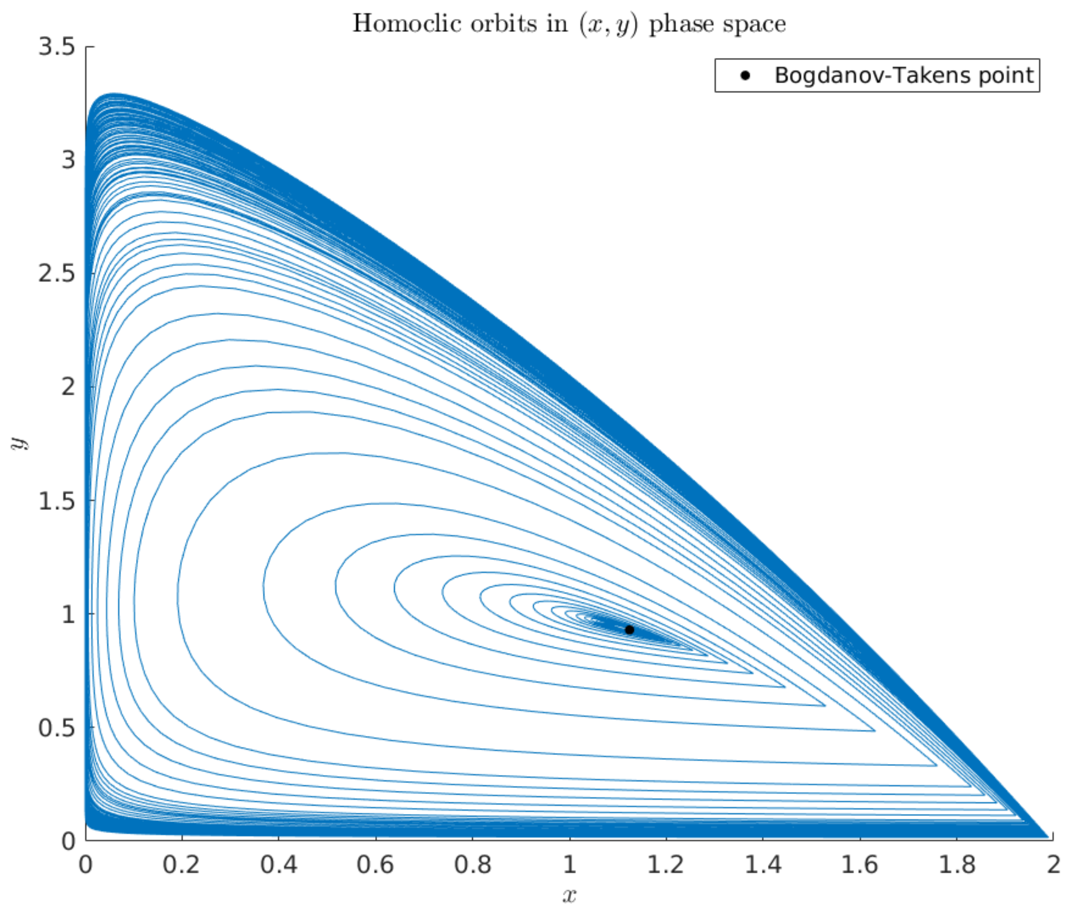
```
%plot --width 1024 --height 800
hold on
global homds
% plot computed parameter curve
plot(homoclinic_br(homds.PeriodIdx+1,:), ...
     homoclinic_br(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent normal form coefficients
bt = BT_nmfm_orbital(odefile, bt, ap, BToptions);
a   = bt.nmfm.a;
b   = bt.nmfm.b;
K10 = bt.nmfm.K10;
K01 = bt.nmfm.K01;
K02 = bt.nmfm.K02;
K11 = bt.nmfm.K11;
K03 = bt.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 0.4);
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alphaSecondOrder = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2;
alpha = bt.par(ap) + alpha;
alphaSecondOrder = bt.par(ap) + alphaSecondOrder;
alpha2016 = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2;
alpha2016 = bt.par(ap) + alpha2016;
% plot currect predictor
plot(alphaSecondOrder(1,:), alphaSecondOrder(2,:), '.', 'MarkerSize', 7)
plot(alpha2016(1,:), alpha2016(2,:), '.', 'MarkerSize', 10)
plot(alpha(1,:), alpha(2,:), '-', 'MarkerSize', 10)
% plot Bogdanov-Takens point
plot(bt.par(ind.d), bt.par(ind.h), '.k', 'MarkerSize', 20)
% set labels and legend
xlabel('$d$')
ylabel('$h$')
legend({'Homoclinic curve', 'Second order homoclinc predictor', ...
    'Second order homoclinic predictor 2016', ...
    'Thrid order homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'SouthWest')
title('Comparision between computed and predicted parameter curve.')
axis([0.1898    0.2596    0.2169    0.3137])
```



6.7 Bifurcation diagram in (x, y) phase-space

To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (x, y) phase-space.

```
hold on
plot(homoclinic_br(homds.coords(1:homds.nphase:end), 1:end), ...
     homoclinic_br(homds.coords(2:homds.nphase:end), 1:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$x$')
ylabel('$y$')
plot(bt.x(1), bt.x(2), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'NorthEast')
title('Homoclic orbits in $(x,y)$ phase space')
```



6.7.1 Predictors of orbits for various epsilons

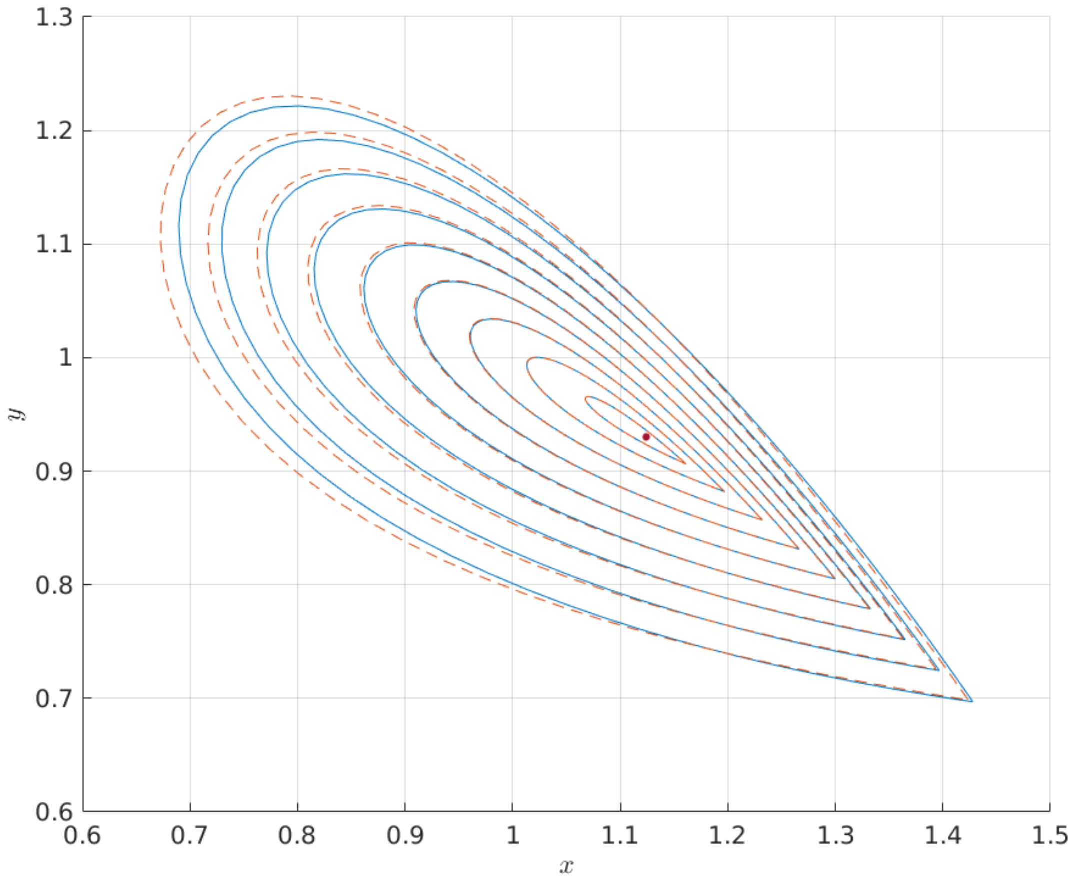
Below we compute for a large range of amplitudes the predicted and corrected homoclinic solutions and compare them in phase space. We see that even with an amplitude of `1` the predicted homoclinic orbit is still very close.

```
options = BT_Hom_set_options();
options.messages = false;
options.correct = false;
options.TTolerance = 1.0e-05;

amplitudes = linspace(1.0e-03, 1,10);
XPredicted = zeros(330,length(amplitudes));
XCorrected = zeros(330,length(amplitudes));
for j=1:length(amplitudes)
  options.amplitude = amplitudes(j);
  [x_pred, v0] = init_BT_Hom(odefile, bt, ap, options);
  XPredicted(:,j) = x_pred;
  try
    XCorrected(:,j) = newtcorr(x_pred, v0);
  catch
    warning('Didn''t convergence to homoclinic solution')
  end
```

(continues on next page)

(continued from previous page)

```
end

hold on
cm = lines;
plot(XPredicted(homds.coords(1:homds.nphase:end),1:10), ...
     XPredicted(homds.coords(2:homds.nphase:end),1:10), ...
      'color', cm(1,:))
plot(XCorrected(homds.coords(1:homds.nphase:end),1:10), ...
     XCorrected(homds.coords(2:homds.nphase:end),1:10), ...
      '--', 'color', cm(2,:))
plot(bt.x(1), bt.x(2), '.', 'MarkerSize', 16)
xlabel('$x$')
ylabel('$y$')
grid on
```



6.8 Convergence plot

We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

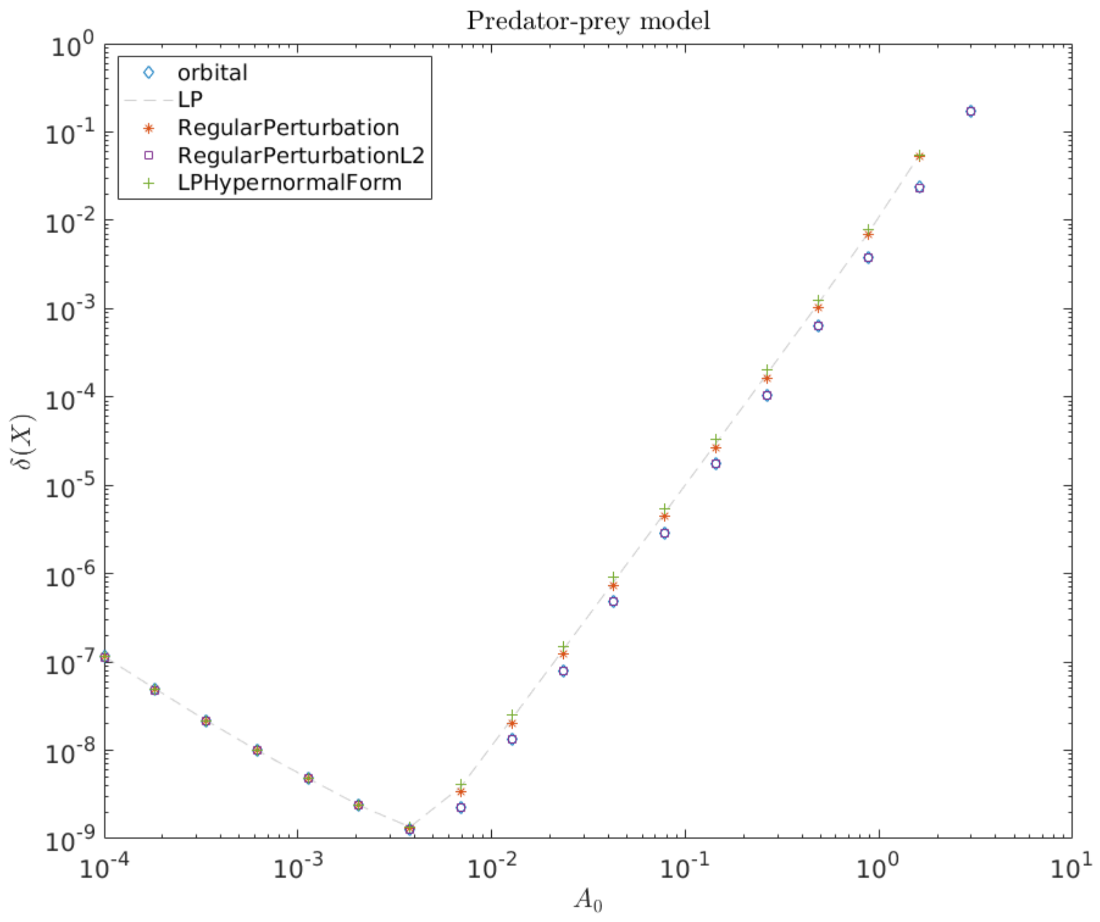
```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 1, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    BToptions.method = methodList{i};
    relativeErrors{i} = zeros(size(amplitudes));
    for j=1:length(amplitudes)
    BToptions.amplitude = amplitudes(j);
    [x_pred, v0] = init_BT_Hom(odefile, bt, ap, BToptions);
    try
        x_corrected = newtcorr(x_pred, v0);
        relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
    catch
        warning('Did not converge.')
        continue
    end
  end
end

cm = lines();
loglog(amplitudes, relativeErrors{1}(:), 'd', ...
       amplitudes, relativeErrors{2}(:), '--', ...
       amplitudes, relativeErrors{3}(:), '*', ...
       amplitudes, relativeErrors{4}(:), 's', ...
       amplitudes, relativeErrors{5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('Predator-prey model')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```

```
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
```

(continues on next page)

(continued from previous page)

```
Warning: Did not converge.
Warning: Did not converge.
```



Part IV

Bazykin

CHAPTER

SEVEN

GENERATE SYSTEM FILES FOR THE BAZYKIN'S PREDATOR-PREY SYSTEM

This Jupyter Notebook generates the **system files** for the following predator-prey ecosystem

$$\begin{cases} \dot{x}_1 = x_1 - \dfrac{x_1 x_2}{1+\alpha x_1} - \epsilon x_1^2, \\ \dot{x}_2 = -\gamma x_2 + \dfrac{x_1 x_2}{1+\alpha x_1} - \delta x_2^2. \end{cases}$$

These are used in the *Bazykin's predator-prey demo*.

7.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

7.2 Define the system name

```
system_name = 'Bazykin';
```

7.3 Create coordinates and parameter names as strings

```
coordsnames = {'x1', 'x2'};
parnames = {'alpha', 'delta'};
```

7.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `alpha` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});        % create symbol for alpha and delta
par=cell2sym(parnames);   % now alpha1 is par(1) etc
syms(coordsnames{:});     % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

7.5 Define fixed parameters

```
gamma = 1;
epsilon = 0.01;
```

7.6 Define the system

```
dx1_dt = x1 - x1*x2/(1+alpha*x1) - epsilon*x1^2;
dx2_dt = -gamma*x2 + x1*x2/(1+alpha*x1) - delta*x2^2;
system = [dx1_dt; dx2_dt];
```

In general there are no modifications needed after this line.

7.7 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

7.8 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

EIGHT

BAZYKIN'S PREDATOR-PREY ECOSYSTEM

This notebook demonstrates initialization of homoclinic orbits emanating form a generic codimension two Bogdanov-Takens bifurcation point in the following predator-prey ecosystem

$$\begin{cases} \dot{x}_1 = x_1 - \dfrac{x_1 x_2}{1+\alpha x_1} - \epsilon x_1^2, \\ \dot{x}_2 = -\gamma x_2 + \dfrac{x_1 x_2}{1+\alpha x_1} - \delta x_2^2. \end{cases}$$

The variables x_1 and x_2 are (scaled) population numbers of prey and predator, respectively, while $\alpha, \gamma, \varepsilon$, and δ are nonnegative parameters describing the behavior of isolated populations and their interaction, see [Baz85, Kuz04]

8.1 Overview

In this demo we will

- Define an analytically derived Bogdanov-Takens point.
- Start continuation of the homoclinic branch emanating from the Bogdanov-Takens point in two parameters (d, h) using the new homoclinic smooth orbital predictor from [Kuz21].
- Compare the predicted and computed homoclinic bifurcation curve emanating from the Bogdanov-Takens point in parameters space.
- Compare a range of predictors for the homoclinic solutions emanating from the Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space.
- Create a convergence plot of the different homoclinic approximations derived in [Kuz21].

8.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Equilibrium'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitPointCycle'])
```

(continues on next page)

(continued from previous page)

```
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
```

8.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *Generate system files for the Bazykin's predator-prey system*.

```
odefile=@Bazykin;
```

8.4 Define Bogdanov-Takens point manually

It is easy to analytically derive a Bogdanov-Takens point at

$$(x_1, x_2) \approx (6.265765528962353, 3.601553936836365),$$

with parameter values $(\alpha, \delta) \approx (0.4536243277781295, 0.1751275929502174)$.

```
bt.x = [6.265765528962353; 3.601553936836365];
bt.par = [0.4536243277781295; 0.1751275929502174];
```

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in *DDE-BifTool*.

```
parnames = {'alpha', 'delta'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
ap = [ind.alpha ind.delta]; % continuation parameters
```

8.5 Continue homoclinic curve emanating from Bogdanov-Takens point

To continue the homoclinic curve emanating from the Bogdanov-Takens point we use the functions `BT_Hom_set_options` and `init_BT_Hom` to obtain an initial approximation to the homoclinic solution (`hom_x`) as well as a tangent vector to the discretized homoclinic solution (`hom_v`) as described in *Initial prediction of homoclinic orbit near Bogdanov-Takens point 1*.

```
opt = contset;
opt.Singularities = 0;
opt.MaxNumPoints = 300;
opt.MaxStepsize = 1;
```

(continues on next page)

(continued from previous page)

```
options = BT_Hom_set_options();
[hom_x, hom_v] = init_BT_Hom(odefile, bt,  ap, options);
homoclinic_br = cont(@homoclinic, hom_x, hom_v, opt);
```

```
Center manifold coefficients' accuracy: 2.664535e-15
BT normal form coefficients:
a=3.945309e-02,         b=-5.253878e-02
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.857576
The initial half-return time T: 55.2395
The initial distance eps0: 0.00153522
The initial distance eps1: 0.000472271
first point found
tangent vector to first point found

elapsed time  = 11.4 secs
npoints curve = 300
```

8.6 Compare predicted with computed parameters

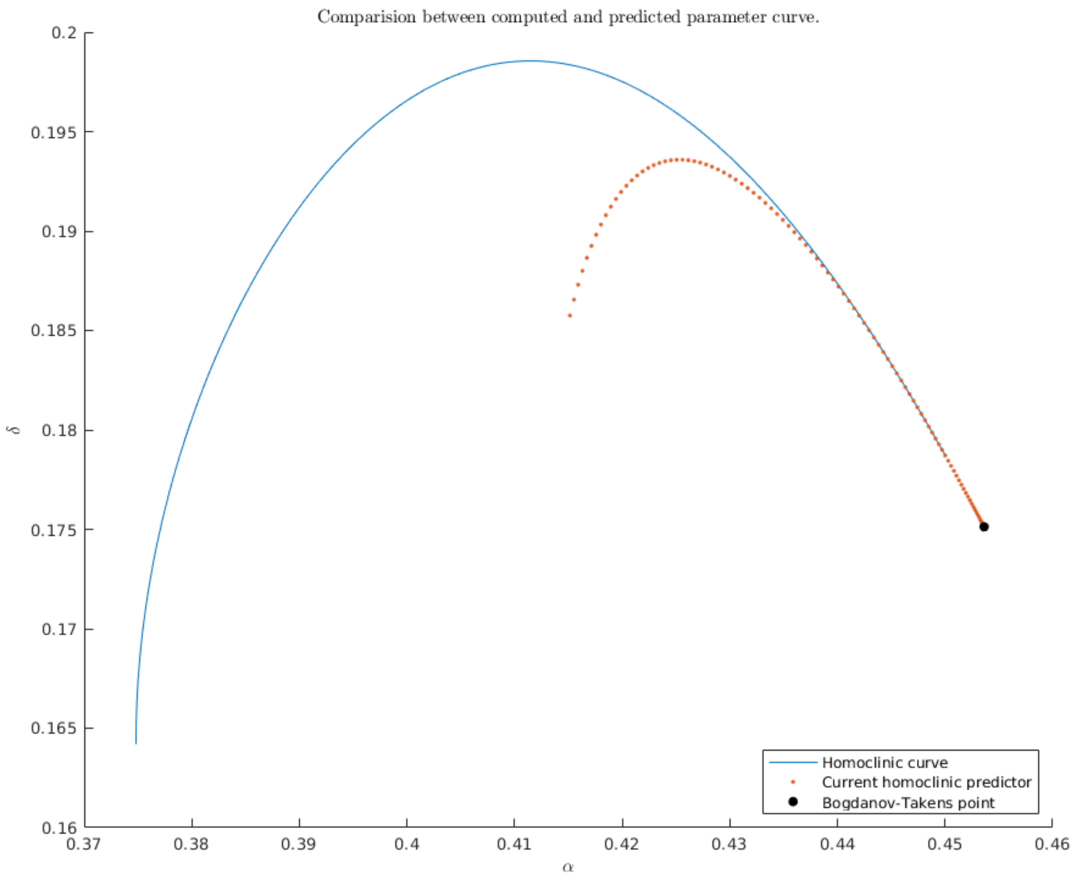
Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm_orbital` to obtain the parameter-dependent normal form coefficients and the coefficients between the parameters of the system and the parameters on the center manifold, see [Kuz21].

```
%plot --width 1024 --height 800
hold on
global homds
% plot computed parameter curve
plot(homoclinic_br(homds.PeriodIdx+1,:), ...
     homoclinic_br(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent normal form coefficients
bt = BT_nmfm_orbital(odefile, bt, ap, BToptions);
a   = bt.nmfm.a;
b   = bt.nmfm.b;
K10 = bt.nmfm.K10;
K01 = bt.nmfm.K01;
K02 = bt.nmfm.K02;
K11 = bt.nmfm.K11;
K03 = bt.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 0.4);
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alpha = bt.par(ap) + alpha;
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.')
% plot Bogdanov-Takens point
plot(bt.par(ind.alpha), bt.par(ind.delta), '.k', 'MarkerSize', 20)
```

(continues on next page)

(continued from previous page)

```
% set axis labels and legend
xlabel('$\alpha$')
ylabel('$\delta$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'SouthEast')
%axis([1.1334    1.1658    0.6305    0.7339])
title('Comparision between computed and predicted parameter curve.')
```



8.7 Bifurcation diagram in phase-space (x_1, x_2)

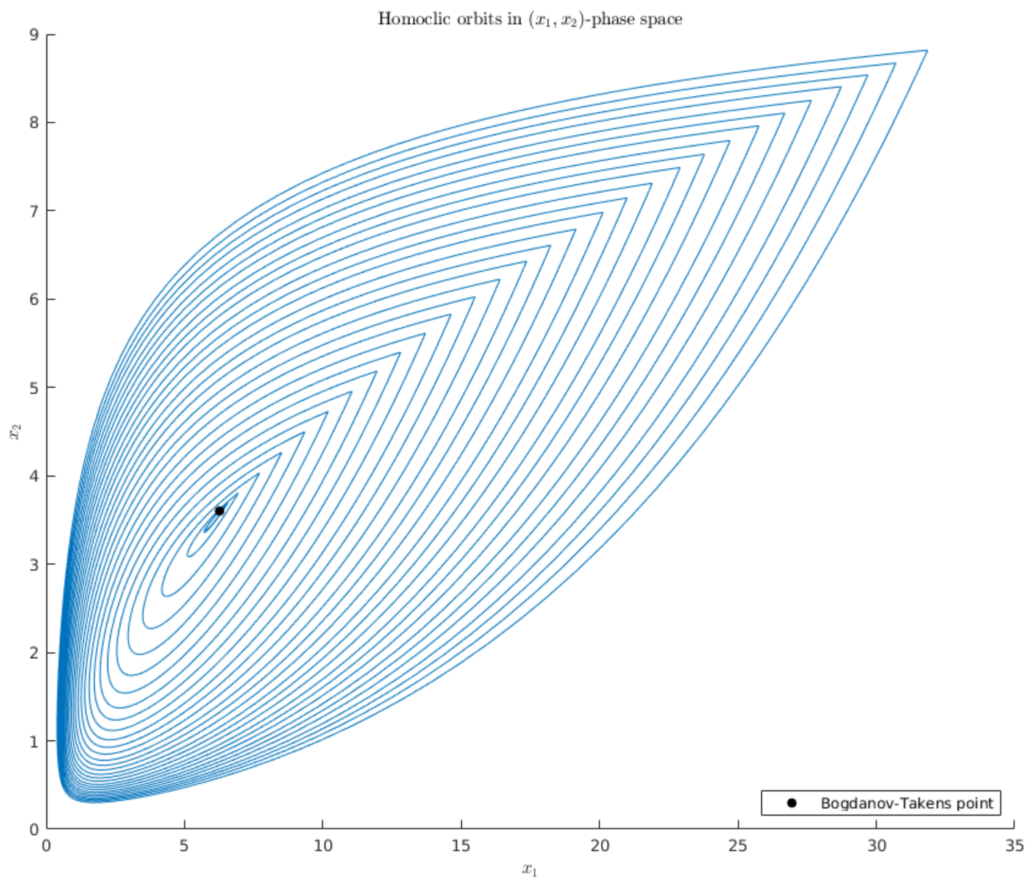
To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (x, y) phase-space.

```
hold on
plot(homoclinic_br(homds.coords(1:homds.nphase:end), 1:10:end), ...
     homoclinic_br(homds.coords(2:homds.nphase:end), 1:10:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$x_1$')
ylabel('$x_2$')
plot(bt.x(1), bt.x(2), '.k' ,'MarkerSize', 20)
```

(continues on next page)

(continued from previous page)

```
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclinic orbits in $(x_1,x_2)$-phase space')
```



8.7.1 Predictors of orbits for various epsilons

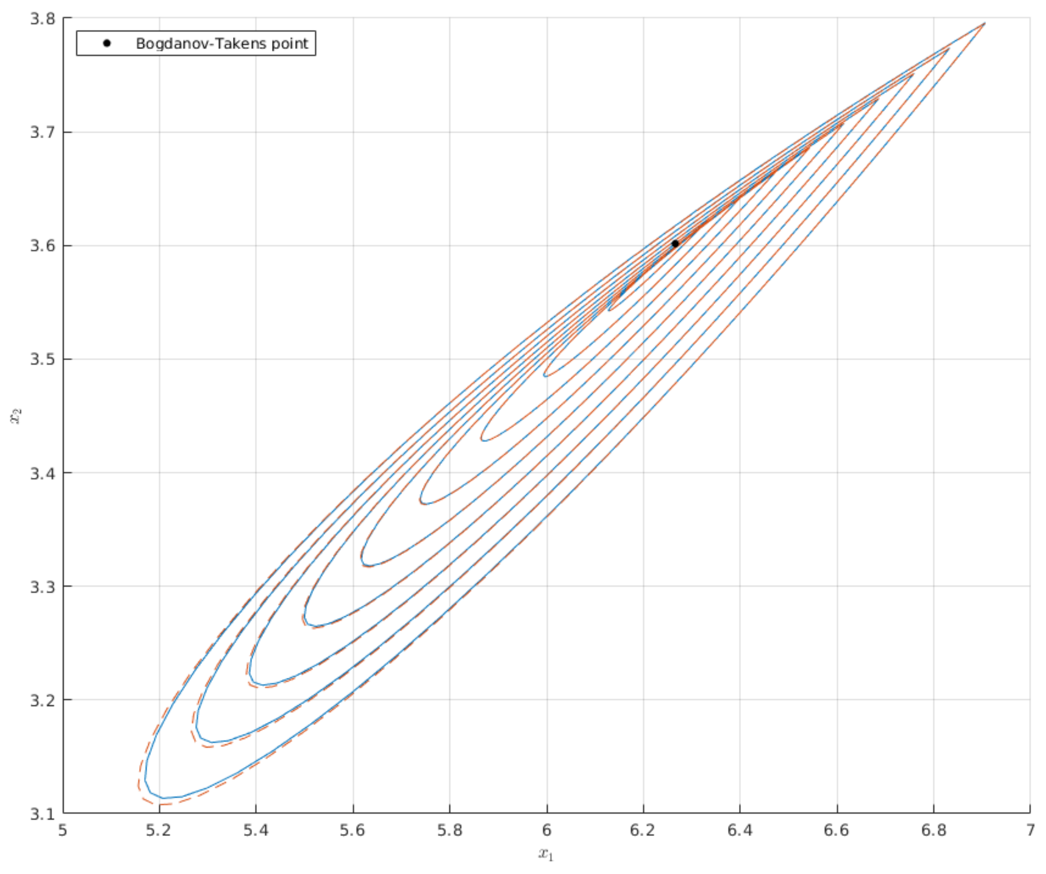
Below we compute for a large range of amplitudes the predicted and corrected homoclinic solutions and compare them in phase space. We see that even with an amplitude of 2 the predicted homoclinic orbit is still very close.

```
options = BT_Hom_set_options();
options.messages = false;
options.correct = false;
options.TTolerance = 1.0e-05;

amplitudes = linspace(1.0e-03, 2,10);
XPredicted = zeros(330,length(amplitudes));
XCorrected = zeros(330,length(amplitudes));
for j=1:length(amplitudes)
  options.amplitude = amplitudes(j);
  [x_pred, v0] = init_BT_Hom(odefile, bt, ap, options);
  XPredicted(:,j) = x_pred;
  try
```

(continues on next page)

(continued from previous page)

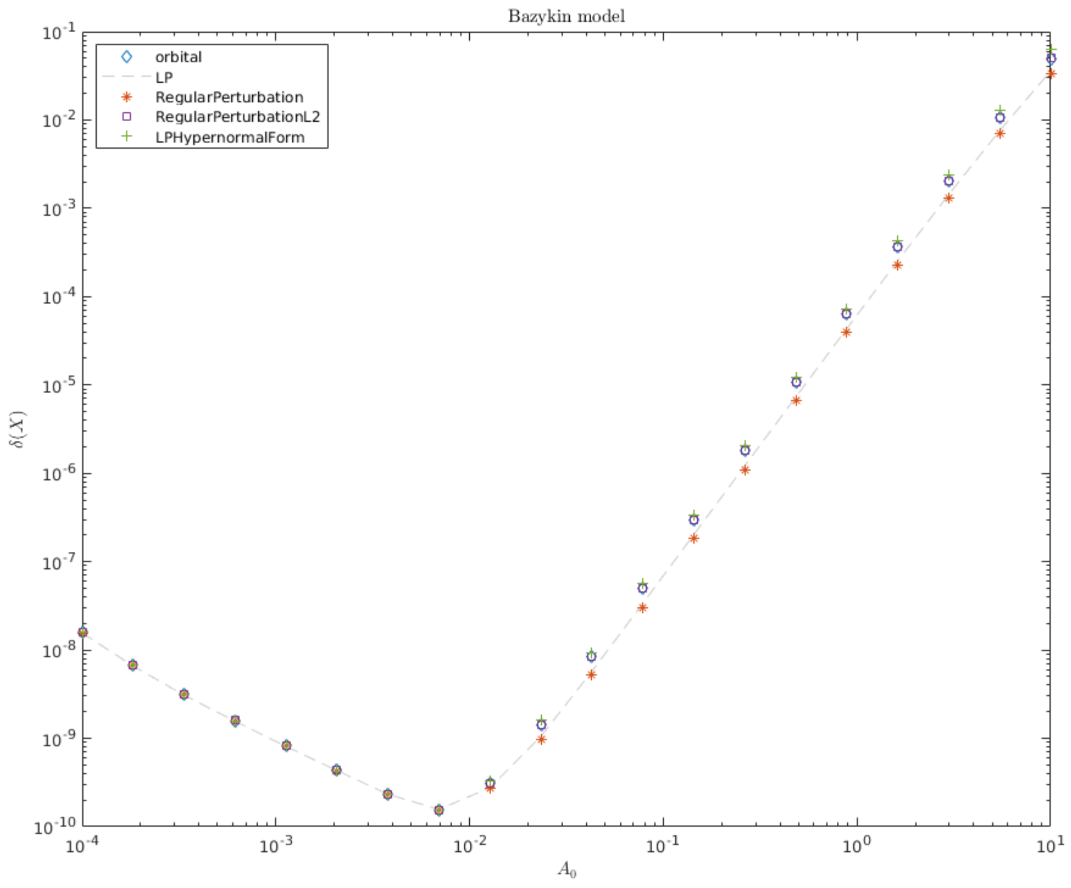
```
    XCorrected(:,j) = newtcorr(x_pred, v0);
  catch
    warning('Didn''t convergence to homoclinic solution')
  end
end

hold on
cm = lines;
plot(XPredicted(homds.coords(1:homds.nphase:end),1:10), ...
     XPredicted(homds.coords(2:homds.nphase:end),1:10), ...
      'color', cm(1,:), 'HandleVisibility', 'Off')
plot(XCorrected(homds.coords(1:homds.nphase:end),1:10), ...
     XCorrected(homds.coords(2:homds.nphase:end),1:10), ...
      '--', 'color', cm(2,:), 'HandleVisibility', 'Off')
plot(bt.x(1), bt.x(2), '.k', 'MarkerSize', 16)
xlabel('$x_1$')
ylabel('$x_2$')
legend('Bogdanov-Takens point', 'Location', 'NorthWest')
grid on
```



8.8 Convergence plot

We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 1, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    BToptions.method = methodList{i};
    relativeErrors{i} = zeros(size(amplitudes));
    for j=1:length(amplitudes)
    BToptions.amplitude = amplitudes(j);
    [x_pred, v0] = init_BT_Hom(odefile, bt, ap, BToptions);
    try
        x_corrected = newtcorr(x_pred, v0);
        relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
    catch
        warning('Did not converge.')
        continue
    end
  end
end

cm = lines();
loglog(amplitudes, relativeErrors{1}(:), 'd', ...
       amplitudes, relativeErrors{2}(:), '--', ...
       amplitudes, relativeErrors{3}(:), '*', ...
       amplitudes, relativeErrors{4}(:), 's', ...
       amplitudes, relativeErrors{5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('Bazykin model')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```



Bazykin model
orbital
LP
RegularPerturbation
RegularPerturbationL2
LPHypernormalForm
$\delta(X)$
A_0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}

Part V

IFOC

CHAPTER

NINE

GENERATE SYSTEM FILES FOR THE INDIRECT FIELD ORIENTED CONTROL SYSTEM

This Jupyter Notebook generates the **system files** for the indirect field oriented control system given by

$$\begin{cases} \dot{x}_1 = -c_1 x_1 + c_2 x_4 - \dfrac{k c_1}{u_2^0} x_2 x_4, \\ \dot{x}_2 = -c_1 x_2 + c_2 u_2^0 + \dfrac{k c_1}{u_2^0} x_1 x_4, \\ \dot{x}_3 = -c_3 x_3 - c_4 c_5 (x_2 x_4 - u_2^0 x_1) + (c_4 T_m + c_3 w_{ref}), \\ \dot{x}_4 = -(k_i - k_p) x_3 - k_p c_4 c_5 (x_2 x_4 - u_2^0 x_1) + k_p (c_4 T_m + c_3 w_{ref}). \end{cases}$$

Here x_1, x_2, x_3 and x_4 are the state variables, where x_1 and x_2 represent, respectively, direct and quadrature components of the rotor flux; x_3 is the rotor speed error; and x_4 denotes the quadrature axis component of the stator current, respectively. We also define the following constants and parameters: u_2^0 is a constant reference for the rotor flux magnitude; c_1 to c_5 are machine parameters; k_p and k_i are the proportional (P) and the integral (I) control gains, respectively; w_{ref} is the speed reference; T_m the load torque; k the measure of rotor time constant mismatches.

These are used in the *IFOC* demo.

9.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

9.2 Set the system name

```
system_name = 'IFOC';
```

9.3 Create coordinates and parameter names as strings

```
coordsnames = {'x1', 'x2', 'x3', 'x4'};
parnames={'k', 'Tm'};
```

9.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `k` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});        % create symbol for alpha and delta
par=cell2sym(parnames);   % now alpha1 is par(1) etc
syms(coordsnames{:});     % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

9.5 Define fixed parameters

```
c1 = 4.4868;
c2 = 0.3567;
c3 = 0;
c4 = 9.743;
c5 = 1.911;
u20 = 11.3;
kp = 4.5;
ki = 500;
wref = 0;
```

9.6 Define the system

```
dx1_dt = -c1*x1 + c2*x4 - k*c1/u20*x2*x4;
dx2_dt = -c1*x2 + c2*u20 + k*c1/u20*x1*x4;
dx3_dt = -c3*x3 - c4*c5*(x2*x4 - u20*x1) + (c4*Tm + c3*wref);
dx4_dt = -(ki-kp*c3)*x3 - kp*c4*c5*(x2*x4 - u20*x1) + kp*(c4*Tm + c3*wref);
system = [dx1_dt; dx2_dt; dx3_dt; dx4_dt];
```

In general there are no modifications needed after this line.

9.7 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
```

(continues on next page)

(continued from previous page)

```
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

9.8 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

TEN

INDIRECT FIELD ORIENTED CONTROL SYSTEM

In [BR00, SGAR08] the following model is considered

$$\begin{cases} \dot{x}_1 = -c_1 x_1 + c_2 x_4 - \dfrac{k c_1}{u_2^0} x_2 x_4, \\ \dot{x}_2 = -c_1 x_2 + c_2 u_2^0 + \dfrac{k c_1}{u_2^0} x_1 x_4, \\ \dot{x}_3 = -c_3 x_3 - c_4 c_5 (x_2 x_4 - u_2^0 x_1) + (c_4 T_m + c_3 w_{ref}), \\ \dot{x}_4 = -(k_i - k_p) x_3 - k_p c_4 c_5 (x_2 x_4 - u_2^0 x_1) + k_p (c_4 T_m + c_3 w_{ref}). \end{cases} \tag{1}$$

Here x_1, x_2, x_3 and x_4 are the state variables, where x_1 and x_2 represent, respectively, direct and quadrature components of the rotor flux; x_3 is the rotor speed error; and x_4 denotes the quadrature axis component of the stator current, respectively. We also define the following constants and parameters: u_2^0 is a constant reference for the rotor flux magnitude; c_1 to c_5 are machine parameters; k_p and k_i are the proportional (P) and the integral (I) control gains, respectively; w_{ref} is the speed reference; T_m the load torque; k the measure of rotor time constant mismatches.

The parameters $c_1, c_2, c_3, c_4, c_5, u_{20}, k_p, k_i$ and w_{ref} are fixed as follows

$$\begin{aligned} c_1 &= 4.4868, \\ c_2 &= 0.3567, \\ c_3 &= 0, \\ c_4 &= 9.743, \\ c_5 &= 1.911, \\ u_2^0 &= 11.3, \\ k_p &= 4.5, \\ k_i &= 500, \\ w_{ref} &= 0, \end{aligned}$$

while the parameters k and T_m are used as unfolding parameters.

10.1 Overview

In this demo we will

- Define an analytically derived Bogdanov-Takens point.
- Start continuation of the homoclinic branch emanating from the Bogdanov-Takens points in two parameters (k, T_m) using the new homoclinic smooth orbital predictor from [Kuz21].
- Compare the predicted and computed homoclinic bifurcation curve emanating from the Bogdanov-Takens point in parameters space.

- Compare a range of predictors for the homoclinic solutions emanating from the Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space.
- Create a convergence plot of the different homoclinic approximations derived in [Kuz21].

10.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Equilibrium'])
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
```

10.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *IFOCGenSym.ipynb*.

```
odefile=@IFOC;
```

10.4 Define Bogdanov-Takens point manually

Since w_{ref} vanishes the equilibria of (1) are invariant under the transformation

$$(x_1, x_2, x_3, x_4, T_m) \mapsto (-x_1, x_2, -x_3, -x_4, -T_m). \tag{2}$$

Then solving simultaneously for equilibria of (1) with vanishing determinant of the Jacobian with respect to the variables x_1, x_2, x_3 and x_4, and the unfolding parameter T_m yields four solutions. Two solutions are given by

$$\begin{aligned}
x_1^{\pm} &= \frac{c_2\left(k^2 \pm \sqrt{k^4 - 10k^2 + 9} - 1\right)|u_{20}|\sqrt{k^2 \mp \sqrt{k^4 - 10k^2 + 9} - 3}}{4\sqrt{2}c_1(k+1)|k|}, \\
x_2^{\pm} &= \frac{c_2\left(\pm\sqrt{k^4 - 10k^2 + 9} + k(k+4) + 3\right)u_{20}}{4c_1k(k+1)}, \\
x_3 &= 0, \\
x_4^{\mp} &= -\frac{|u_{20}|\sqrt{\left(k^2 \mp \sqrt{k^4 - 10k^2 + 9} - 3\right)}}{\sqrt{2}|k|}, \\
T_m^{\pm} &= -\frac{c_2c_5\left(k^2 \pm \sqrt{k^4 - 10k^2 + 9} + 3\right)u_{20}|u_{20}|\sqrt{k^2 \mp \sqrt{k^4 - 10k^2 + 9} - 3}}{4\sqrt{2}c_1k|k|},
\end{aligned}$$

while the second two follow from the symmetry (2).

Next we solve for a double zero of the characteristic equation of the Jacobian with respect to the unfolding parameter k. This reveals that there are two solutions given by

$$(\pm x_1^-, x_2^-, 0, \pm x_4^+, \pm T_m^-),$$

with

$$k = 1 + \frac{1}{3}\sqrt[3]{216 - 24\sqrt{57}} + \frac{2\sqrt[3]{9+\sqrt{57}}}{3^{2/3}}.$$

Thus we can explicitly define the Bogdanov-Takens point

```
c1 = 4.4868;
c2 = 0.3567;
c5 = 1.911;
u20 = 11.3;
k = 1+(1/3)*(216+(-24)*57^(1/2))^(1/3)+2*3^(-2/3)*(9+57^(1/2))^(1/3);
x1m = (1/4)*2^(-1/2)*c1^(-1)*c2*(1+k)^(-1)*((-1)+k^2+(-1)*(9+( ...
      -10)*k^2+k^4)^(1/2))*(k^(-2)*((-3)+k^2+(9+(-10)*k^2+ ...
      k^4)^(1/2))*u20^2)^(1/2);
x2m = (1/4)*c1^(-1)*c2*k^(-1)*(1+k)^(-1)*(3+k*(4+k)+(-1)*(9+( ...
      -10)*k^2+k^4)^(1/2))*u20;
x3 = 0;
x4p = (-1)*2^(-1/2)*(k^(-2)*((-3)+k^2+(9+(-10)*k^2+k^4)^(1/2))*u20^2)^(1/2);
Tmm = (-1/4)*2^(-1/2)*c1^(-1)*c2*c5*k^(-1)*(3+k^2+(-1)*(9+( ...
      -10)*k^2+k^4)^(1/2))*u20*(k^(-2)*((-3)+k^2+(9+(-10)* ...
      k^2+k^4)^(1/2))*u20^2)^(1/2);
bt1.x = [x1m; x2m; x3; x4p];
bt1.par = [k; Tmm];
```

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in *DDE-BifTool*.

```
parnames = {'k', 'Tm'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
```

10.5 Continue homoclinic curve emanating from the Bogdanov-Takens point

To continue the homoclinic curve emanating from the Bogdanov-Takens point we use the functions `BT_Hom_set_options` and `init_BT_Hom` to obtain an initial approximation to the homoclinic solution (`hom_x`) as well as a tangent vector to the discretized homoclinic solution (`hom_v`) as described in *Initial prediction of homoclinic orbit near Bogdanov-Takens point 1*.

```
ap = [ind.k, ind.Tm];
BToptions = BT_Hom_set_options();
[hom_x, hom_v] = init_BT_Hom(odefile, bt1,  ap, BToptions);
opt = contset;
opt.Singularities = 0;
opt.MaxNumPoints = 900;
homoclinic_br1 = cont(@homoclinic, hom_x, hom_v, opt);
```

```
BT normal form coefficients:
a=-2.800604e+01,         b=9.109991e-01
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 2.02473
The initial half-return time T: 1.17262
The initial distance time eps0: 0.0149101
The initial distance time eps1: 0.00705633
first point found
tangent vector to first point found

elapsed time  = 82.1 secs
npoints curve = 900
```

We see that the initial amplitude is large compared to the previously considered models. Nonetheless the homoclinic approximation converges and the homoclinic solution is continued correctly.

10.6 Compare predicted with computed parameters

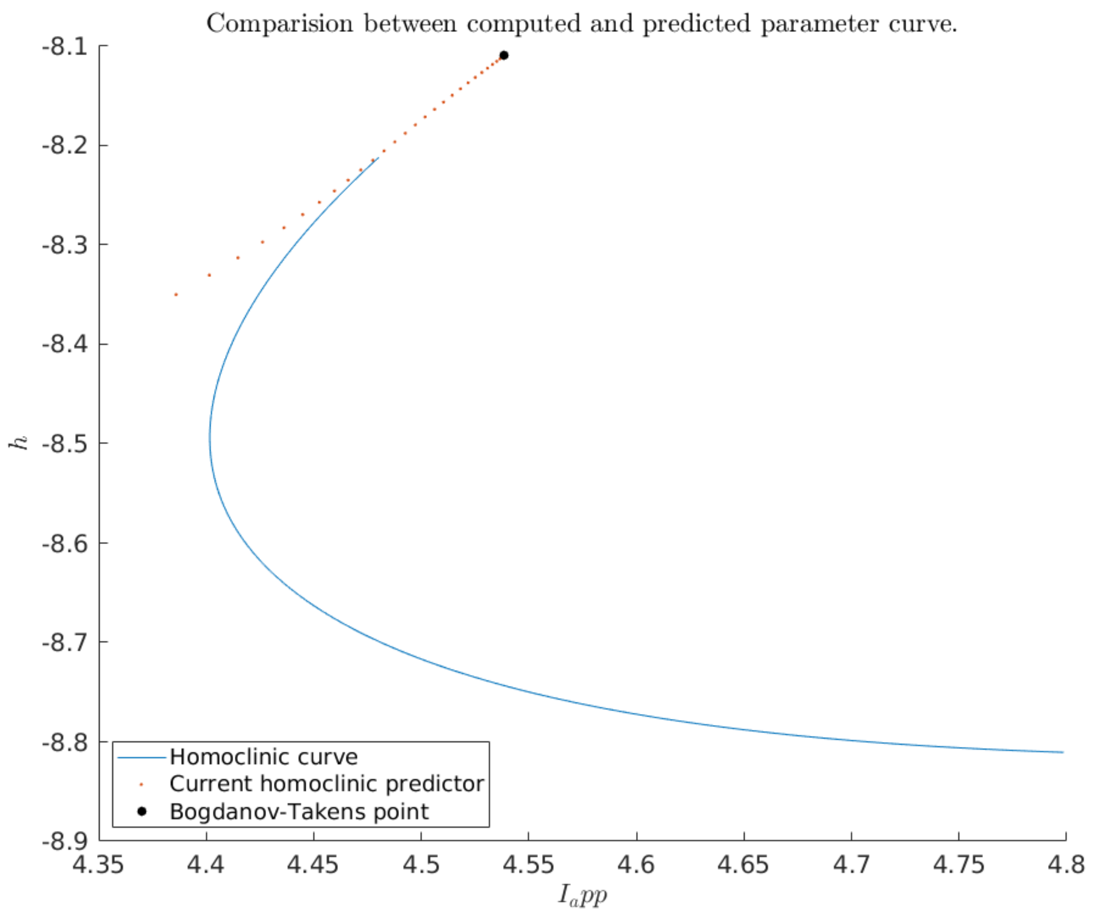
Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm_orbital` to obtain the smooth orbital normal form coefficients, i.e. a and b, and the coefficients of the transformation K between the parameters of the system and the parameters of the smooth orbital normal form on the center manifold, see [Kuz21].

```
%plot --width 1024 --height 800
hold on
global homds
% plot computed parameter curve
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent normal form coefficients
bt1 = BT_nmfm_orbital(odefile, bt1, ap, BToptions);
```

(continues on next page)

(continued from previous page)

```
a   = bt1.nmfm.a;
b   = bt1.nmfm.b;
K10 = bt1.nmfm.K10;
K01 = bt1.nmfm.K01;
K02 = bt1.nmfm.K02;
K11 = bt1.nmfm.K11;
K03 = bt1.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 0.15, 30);
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alpha = bt1.par(ap) + alpha;
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.')
% plot Bogdanov-Takens point
plot(bt1.par(ind.k), bt1.par(ind.Tm), '.k', 'MarkerSize', 20)
% set axis labels and legend
xlabel('$I_app$')
ylabel('$h$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'SouthWest')
title('Comparision between computed and predicted parameter curve.')
```

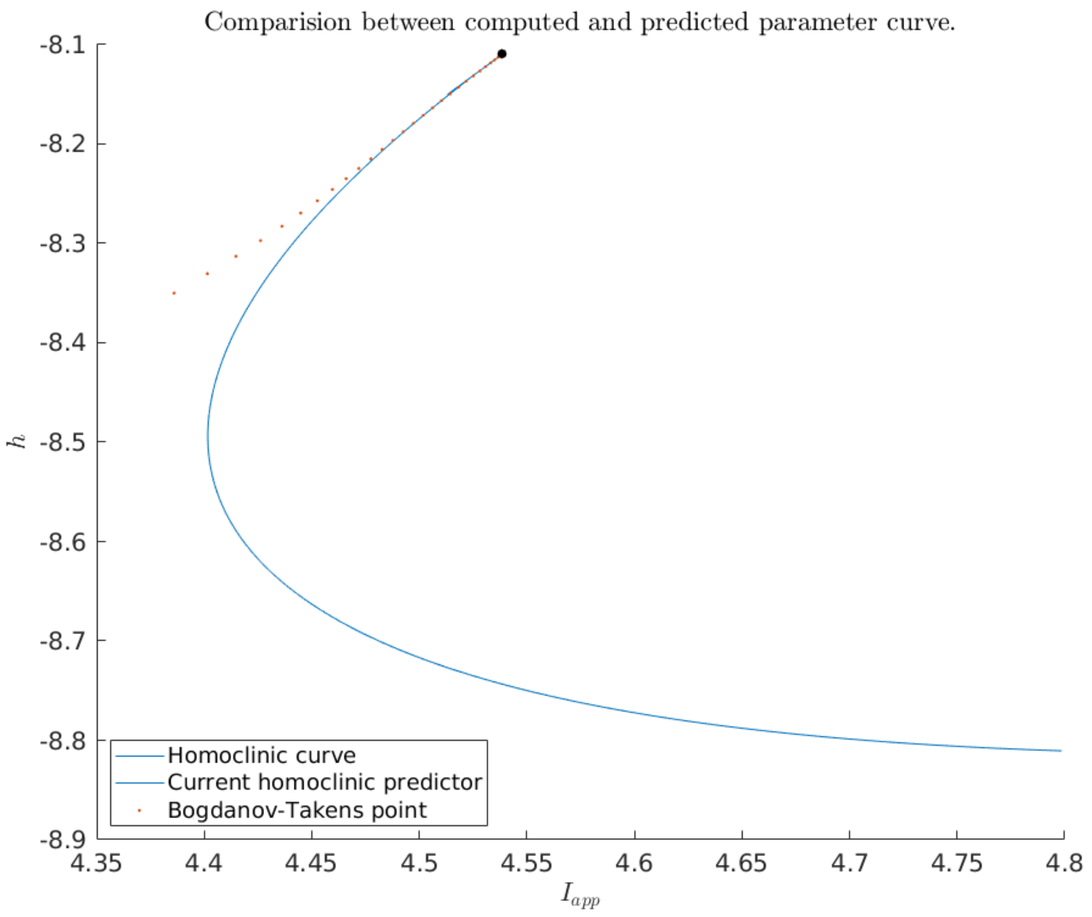


If we would like to have the homoclinic curve close to the Bogdanov-Takens point we could start the continuation of the homoclinic orbit with a smaller amplitude, for example with `BToptions.amplitude = 0.1`. Another possibility is to continue the homoclinic curve in the reverse direction. Here we choose the second approach.

```
[hom_x, hom_v] = init_BT_Hom(odefile, bt1,  ap, BToptions);
opt.Singularities = 0;
opt.MaxNumPoints = 120;
opt.Backward = 1;
homoclinic_br1_rev = cont(@homoclinic, hom_x, hom_v, opt);
```

```
BT normal form coefficients:
a=-2.800604e+01,         b=9.109991e-01
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 2.02473
The initial half-return time T: 1.17262
The initial distance time eps0: 0.0149101
The initial distance time eps1: 0.00705633
first point found
tangent vector to first point found
Current step size too small (point 97)
elapsed time  = 9.9 secs
npoints curve = 97
```

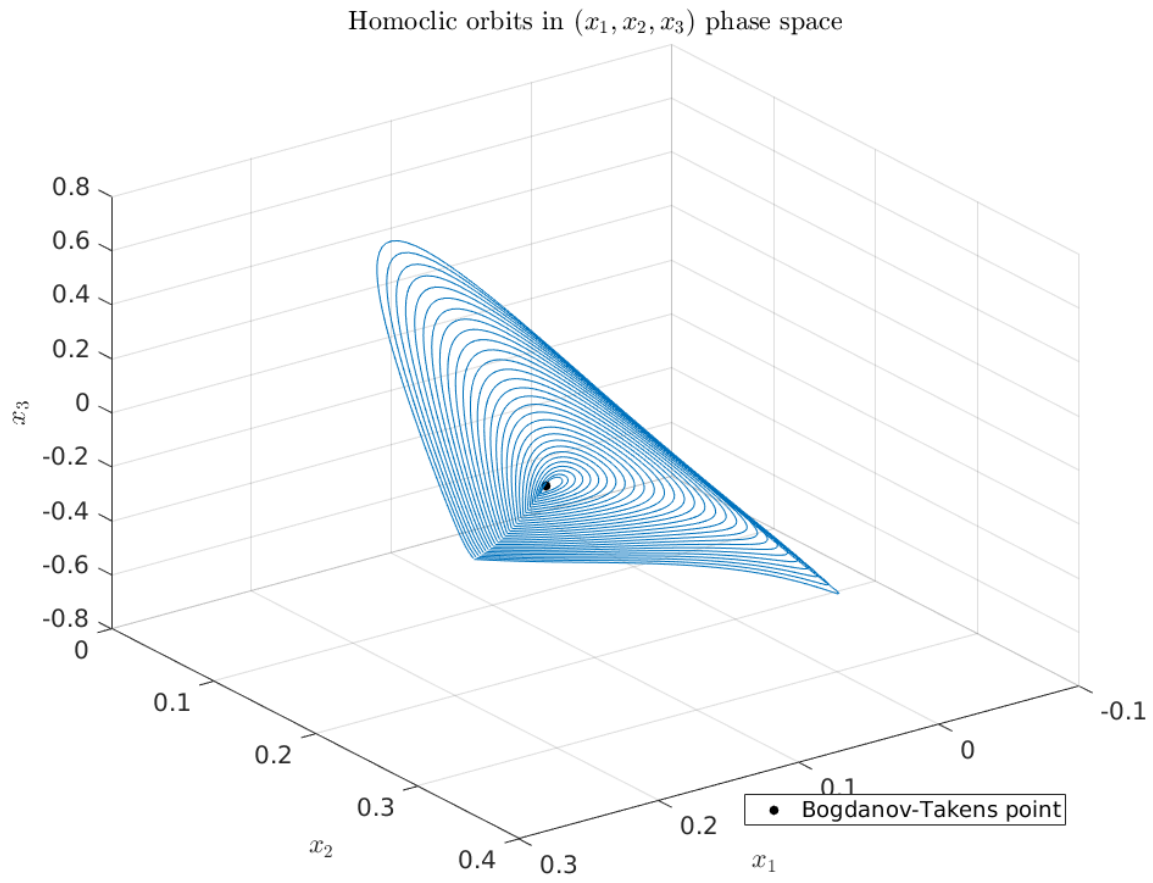
```
%plot --width 1024 --height 800
hold on
cm = lines();
homColor  = cm(1,:);
% plot computed parameter curve
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:), ...
     'Color', homColor);
plot(homoclinic_br1_rev(homds.PeriodIdx+1,:), ...
     homoclinic_br1_rev(homds.PeriodIdx+2,:), ...
     'Color', homColor);
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.', 'Color', cm(2,:))
% plot Bogdanov-Takens point
plot(bt1.par(ind.k), bt1.par(ind.Tm), '.k', 'MarkerSize', 20)
% set axis labels and legend
xlabel('$I_{app}$')
ylabel('$h$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'SouthWest')
title('Comparision between computed and predicted parameter curve.')
```



10.7 Plot of continued homoclinic solutions in (x_1, x_2, x_3) phase-space

To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (x_1, x_2, x_3) phase-space.

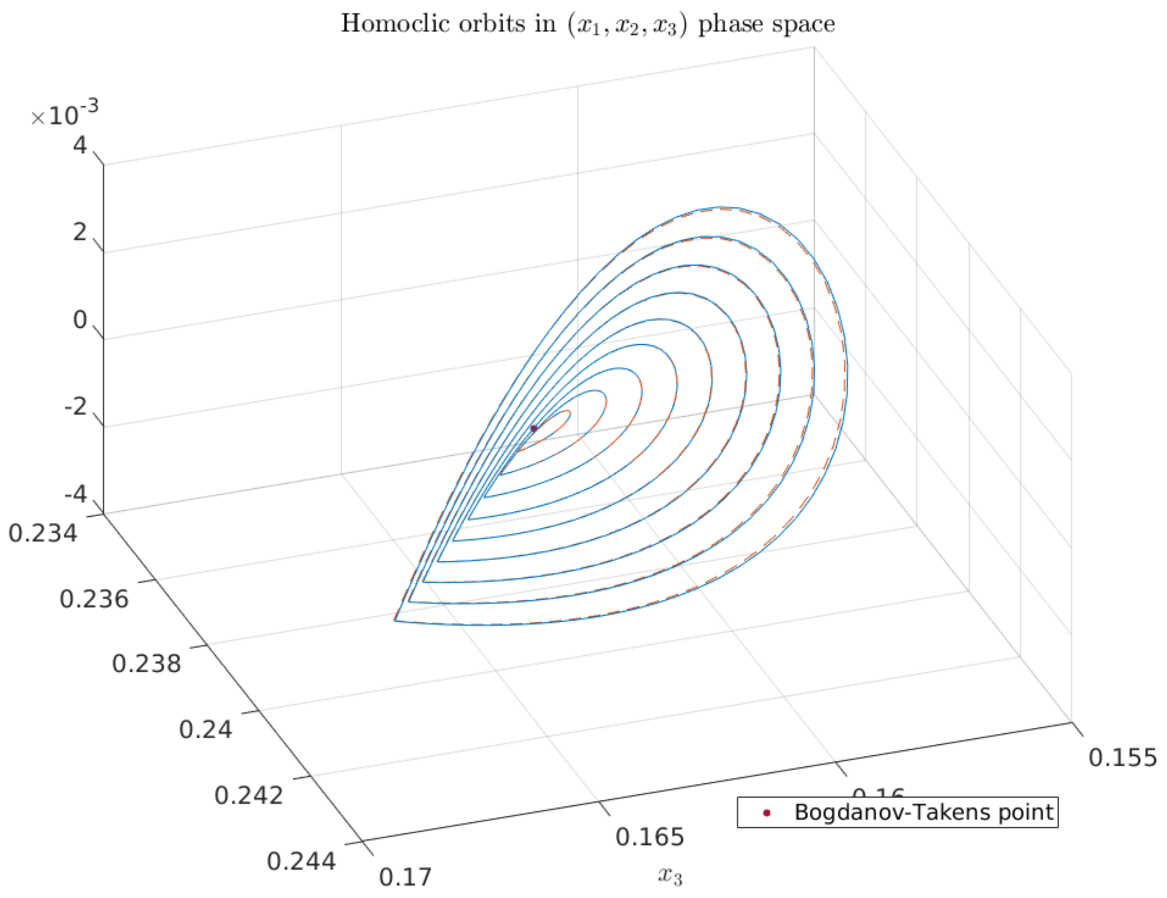
```
hold on
plot3(homoclinic_br1(homds.coords(1:homds.nphase:end), 1:30:end), ...
      homoclinic_br1(homds.coords(2:homds.nphase:end), 1:30:end), ...
      homoclinic_br1(homds.coords(3:homds.nphase:end), 1:30:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$x_1$')
ylabel('$x_2$')
zlabel('$x_3$')
plot3(bt1.x(1), bt1.x(2), bt1.x(3), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(x_1,x_2,x_3)$ phase space')
grid on
view([144, 31]);
```



10.7.1 Predictors of orbits for various epsilons

Below we compute for a large range of amplitudes the predicted and corrected homoclinic solutions and compare them in phase space. We see that even with an amplitude of `1` the predicted homoclinic orbit is still very close.

```
options = BT_Hom_set_options();
options.messages = false;
options.correct = false;
options.TTolerance = 1.0e-05;

amplitudes = linspace(1.0e-03, 1, 10);
XPredicted = zeros(660,length(amplitudes));
XCorrected = zeros(660,length(amplitudes));
for j=1:length(amplitudes)
  options.amplitude = amplitudes(j);
  [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, options);
  XPredicted(:,j) = x_pred;
  try
    XCorrected(:,j) = newtcorr(x_pred, v0);
  catch
    warning('Didn''t convergence to homoclinic solution')
  end
end

hold on
cm = lines;
plot3(XPredicted(homds.coords(1:homds.nphase:end),1:10), ...
      XPredicted(homds.coords(2:homds.nphase:end),1:10), ...
      XPredicted(homds.coords(3:homds.nphase:end),1:10), ...
      'color', cm(1,:), 'HandleVisibility', 'Off')
plot3(XCorrected(homds.coords(1:homds.nphase:end),1:10), ...
      XCorrected(homds.coords(2:homds.nphase:end),1:10), ...
      XCorrected(homds.coords(3:homds.nphase:end),1:10), ...
      '--', 'color', cm(2,:), 'HandleVisibility', 'Off')
plot3(bt1.x(1), bt1.x(2), bt1.x(3), '.', 'MarkerSize', 16)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclinic orbits in $(x_1,x_2,x_3)$ phase space')
xlabel('$x_1$')
xlabel('$x_2$')
xlabel('$x_3$')
grid on
view([160, 45]);
```



10.8 Convergence plot

We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

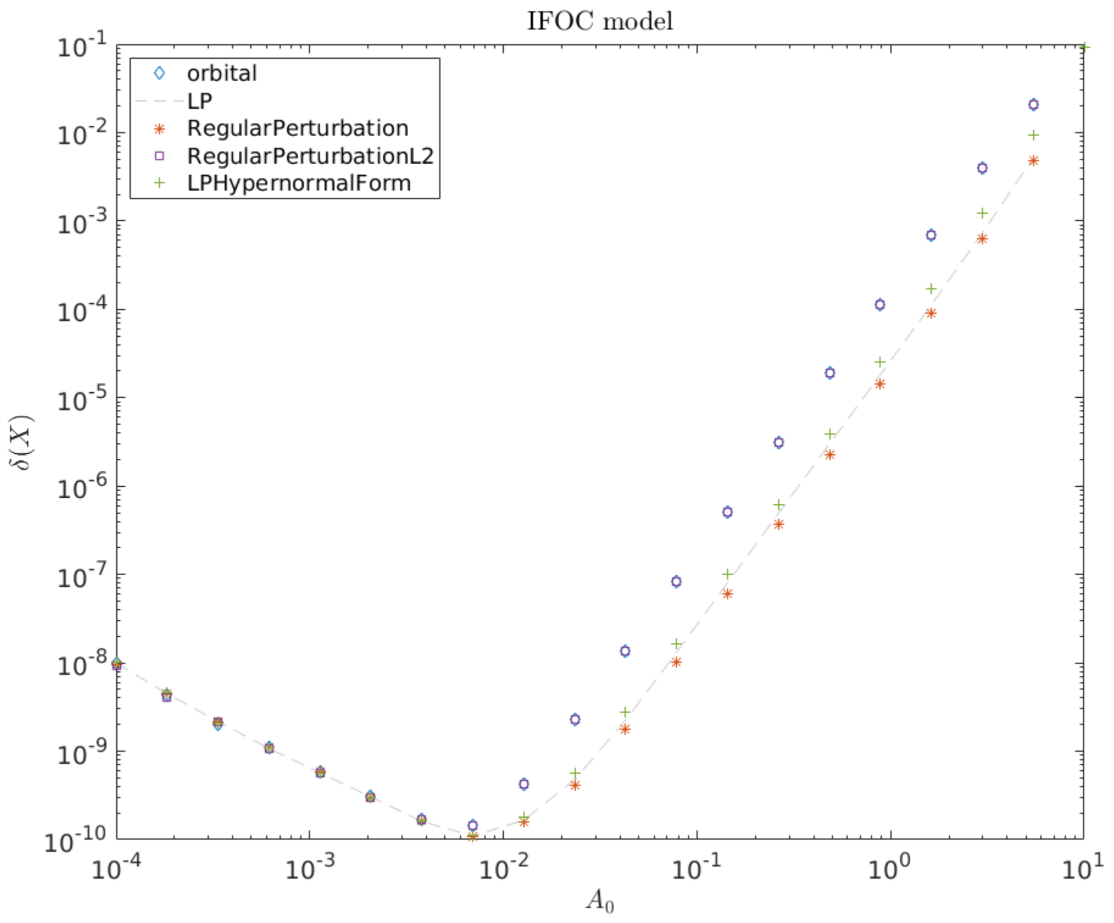
```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 1, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    BToptions.method = methodList{i};
    relativeErrors{i} = zeros(size(amplitudes));
    for j=1:length(amplitudes)
    BToptions.amplitude = amplitudes(j);
```

(continues on next page)

(continued from previous page)

```
    [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, BToptions);
    try
        x_corrected = newtcorr(x_pred, v0);
        relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
    catch
        warning('Did not converge.')
        continue
    end
  end
end

cm = lines();
loglog(amplitudes, relativeErrors{1}(:), 'd', ...
       amplitudes, relativeErrors{2}(:), '--', ...
       amplitudes, relativeErrors{3}(:), '*', ...
       amplitudes, relativeErrors{4}(:), 's', ...
       amplitudes, relativeErrors{5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('IFOC model')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```

```
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
```


IFOC model
orbital
LP
RegularPerturbation
RegularPerturbationL2
LPHypernormalForm
$\delta(X)$
A_0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}

Part VI

Extended Lorenz 84

CHAPTER

ELEVEN

GENERATE SYSTEM FILES FOR THE EXTENDED LORENZ-84 MODEL

This Jupyter Notebook generates the **system files** for the extended Lorenz-84 model given by

$$\begin{cases} \dot{X} = -Y^2 - Z^2 - \alpha X + \alpha F - \xi U^2 \\ \dot{Y} = XY - \beta XZ - Y + G \\ \dot{Z} = \beta XY + XZ - Z \\ \dot{U} = -\delta U + \xi UX + S \end{cases}$$

In this system, X models the intensity of a baroclinic wave, Y and Z the sin and cos coefficients of the wave respectively, the variable U is added to study the influence of external parameters such as temperature. We fix the parameters as follows $\alpha = 0.25, \beta = 1, G = 0.25, \delta = 1.04$ and $\xi = 0.987$. The continuation parameters are F and S.

These are used in the *extended Lorenz-84 model* demo.

11.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, 'Utilities'])
if isOctave
  pkg load symbolic % for GNU Octave
end
```

11.2 Set the system name

```
system_name = 'extendedLorenz84';
```

11.3 Create coordinates and parameter names as strings

```
coordsnames = {'X', 'Y', 'Z', 'U'};
parnames = {'F', 'S'};
```

11.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `alpha` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});        % create symbol for alpha and delta
par=cell2sym(parnames);   % now alpha1 is par(1) etc
syms(coordsnames{:});     % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

11.5 Define fixed parameters

```
alpha = 0.25;
beta = 1;
G = 0.25;
delta = 1.04;
xi = 0.987;
```

11.6 Define the system

```
dX_dt = -Y^2-Z^2-alpha*X+alpha*F-xi*U^2;
dY_dt = X*Y-beta*X*Z-Y+G;
dZ_dt = beta*X*Y+X*Z-Z;
dU_dt = -delta*U+xi*U*X+S;
system = [dX_dt; dY_dt; dZ_dt; dU_dt];
```

In general there are no modifications needed after this line.

11.7 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

11.8 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

TWELVE

THE EXTENDED LORENZ-84 MODEL

The extended Lorenz-84 model given by

$$\begin{cases} \dot{X} = -Y^2 - Z^2 - \alpha X + \alpha F - \xi U^2, \\ \dot{Y} = XY - \beta XZ - Y + G, \\ \dot{Z} = \beta XY + XZ - Z, \\ \dot{U} = -\delta U + \xi UX + S. \end{cases} \tag{1}$$

Here X models the intensity of a baroclinic wave, Y and Z the sin and cos coefficients of the wave respectively, the variable U is added to study the influence of external parameters such as temperature. In this model all codimension two bifurcation points of equilibria are present, see [KMGS08, ShilNikovNN95, vV03]. We fix the parameters as follows $\alpha = 0.25, \beta = 1, G = 0.25, \delta = 1.04$ and $\xi = 0.987$. The continuation parameters are F and S.

12.1 Overview

In this demo we will

- Define a numerically derive Bogdanov-Takens point.
- Start continuation of the homoclinic branch emanating from the Bogdanov-Takens points in two parameters (F, S) using the new homoclinic smooth orbital predictor from [Kuz21].
- Compare the predicted and computed homoclinic bifurcation curve emanating from the Bogdanov-Takens point in parameters space.
- Compare a range of predictors for the homoclinic solutions emanating from the Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space.
- Create a convergence plot of the different homoclinic approximations derived in [Kuz21].

12.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
restoredefaultpath
matcontpath = '../../';
addpath(matcontpath)
addpath([matcontpath, 'Equilibrium'])
```

(continues on next page)

(continued from previous page)

```
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
```

12.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *Lorenz84GenSym.ipynb*.

```
odefile=@extendedLorenz84;
```

12.4 Define Bogdanov-Takens point manually

By numerically solving for real equilibria of (1) with double zero eigenvalues we obtain two equilibria. The first equilibrium is given by

$$\begin{pmatrix} X \\ Y \\ Z \\ U \\ F \\ S \end{pmatrix} \approx \begin{pmatrix} 1.225640495986278 \\ -0.036320793271068824 \\ 0.19728832311233313 \\ -0.12339001198518826 \\ 1.446716701115361 \\ 0.020940169683322466 \end{pmatrix},$$

while the second is obtained from the symmetry [(X,Y,Z,U,F,S) \mapsto (X,Y,Z,-U,F,-S) .]

Thus we can explicitly define the Bogdanov-Takens point

```
X = 1.225640495986278;
Y = -0.036320793271068824;
Z = 0.19728832311233313;
U = -0.12339001198518826;
F = 1.446716701115361;
S = 0.020940169683322466;
bt1.x = [X; Y; Z; U];
bt1.par = [F; S];
```

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in *DDE-BifTool*.

```
parnames = {'F', 'S'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
```

12.5 Continue homoclinic curve emanating from the Bogdanov-Takens point

To continue the homoclinic curve emanating from the Bogdanov-Takens point we use the functions `BT_Hom_set_options` and `init_BT_Hom` to obtain an initial approximation to the homoclinic solution (`hom_x`) as well as a tangent vector to the discretized homoclinic solution (`hom_v`) as described in *Initial prediction of homoclinic orbit near Bogdanov-Takens point 1*.

```
ap = [ind.F, ind.S];
BToptions = BT_Hom_set_options();
[hom_x, hom_v] = init_BT_Hom(odefile, bt1,  ap, BToptions);
opt = contset;
opt.Singularities = 0;
homoclinic_br1 = cont(@homoclinic, hom_x, hom_v, opt);
```

```
Center manifold coefficients' accuracy: 1.887379e-15
BT normal form coefficients:
a=2.144234e-01,         b=6.065146e-01
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.0349736
The initial half-return time T: 116.985
The initial distance eps0: 1.96422e-05
The initial distance eps1: 6.35711e-05
first point found
tangent vector to first point found
Current step size too small (point 43)
elapsed time  = 2.5 secs
npoints curve = 43
```

12.6 Compare predicted with computed parameters

Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm_orbital` to obtain the smooth orbital normal form coefficients, i.e. a and b, and the coefficients of the transformation K between the parameters of the system and the parameters of the smooth orbital normal form on the center manifold, see [Kuz21].

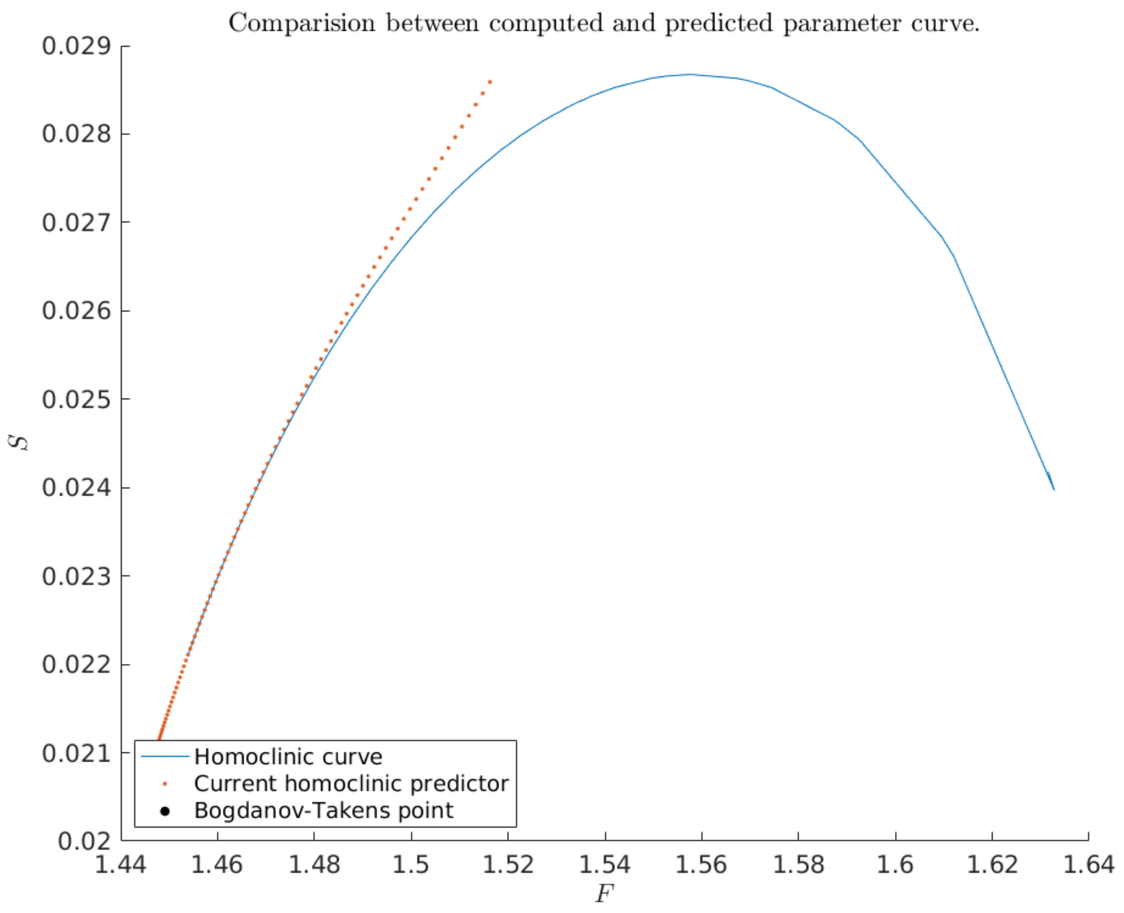
```
%plot --width 1024 --height 800
hold on
global homds
% plot computed parameter curve
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent normal form coefficients
bt1 = BT_nmfm_orbital(odefile, bt1, ap, BToptions);
a   = bt1.nmfm.a;
b   = bt1.nmfm.b;
K10 = bt1.nmfm.K10;
K01 = bt1.nmfm.K01;
K02 = bt1.nmfm.K02;
K11 = bt1.nmfm.K11;
K03 = bt1.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 0.3);
```

(continues on next page)

(continued from previous page)

```
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alpha = bt1.par(ap) + alpha;
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.', 'LineWidth', 8)
% plot Bogdanov-Takens point
plot(bt1.par(ind.F), bt1.par(ind.S), '.k', 'MarkerSize', 20)
% set axis labels and legend
xlabel('$F$')
ylabel('$S$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'SouthWest')
title('Comparision between computed and predicted parameter curve.')
```

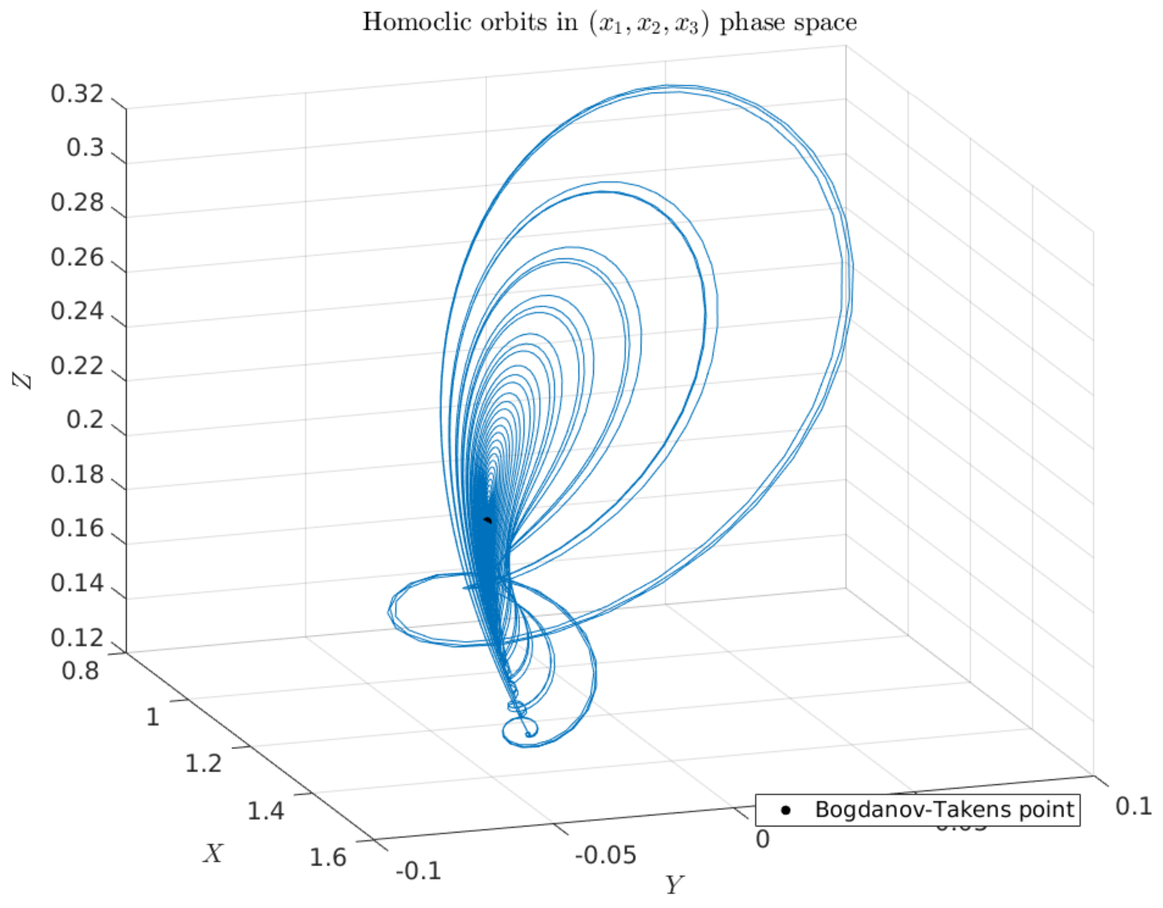
```
Center manifold coefficients' accuracy: 1.887379e-15
```



12.7 Plot of continued homoclinic solutions in (X, Y, Z) phase-space

To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (X, Y, Z) phase-space.

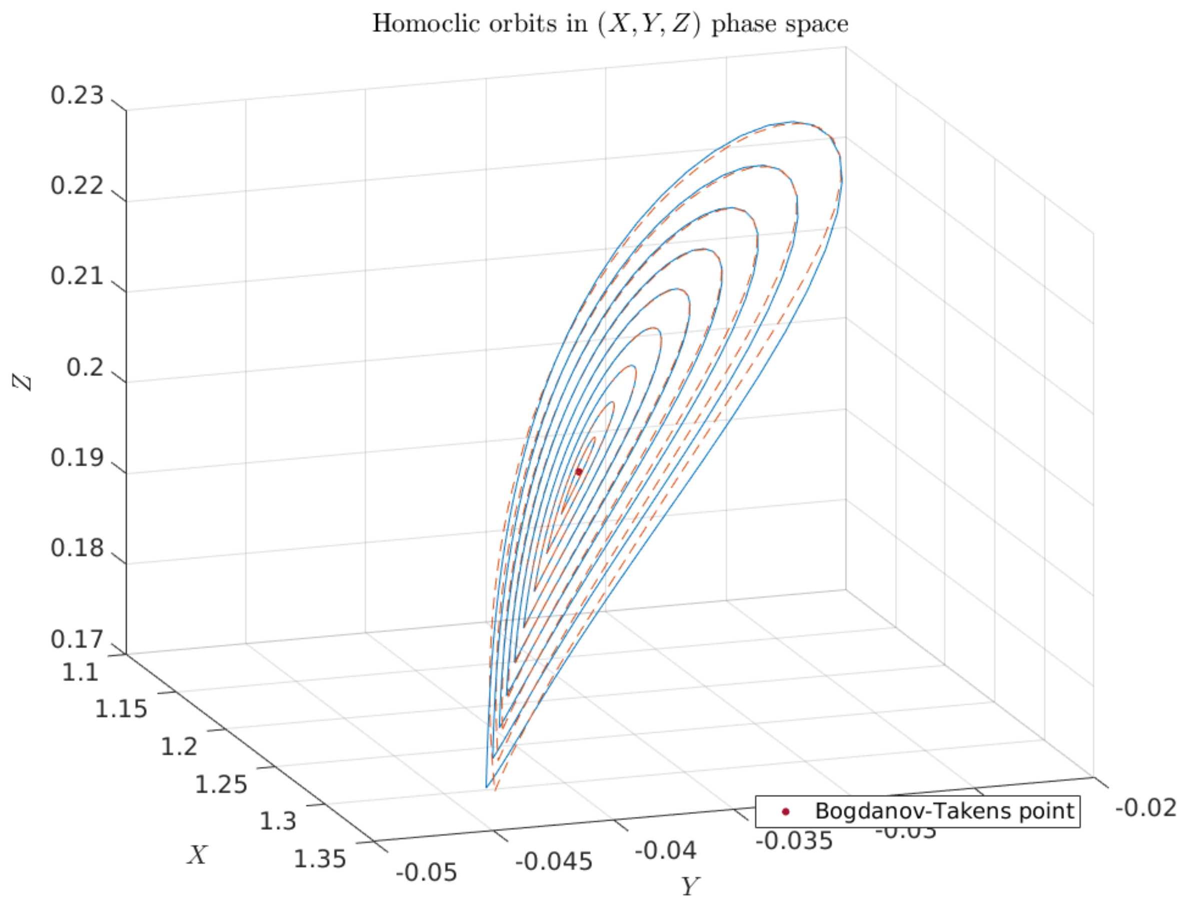
```
hold on
plot3(homoclinic_br1(homds.coords(1:homds.nphase:end), 1:end), ...
      homoclinic_br1(homds.coords(2:homds.nphase:end), 1:end), ...
      homoclinic_br1(homds.coords(3:homds.nphase:end), 1:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$X$')
ylabel('$Y$')
zlabel('$Z$')
plot3(bt1.x(1), bt1.x(2), bt1.x(3), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(x_1,x_2,x_3)$ phase space')
grid on
view([71, 20]);
```



12.7.1 Predictors of orbits for various epsilons

Below we compute for a range of amplitudes the predicted and corrected homoclinic solutions and compare them in phase space. As seen from the plot *above* the homoclinic solutions start to deform rather quickly. Therefore, we cannot expect the approximation to be accurate for relative large amplitude values. Still, for an amplitude of 0.3 the approximation is nearby the corrected homoclinic solutions.

```
options = BT_Hom_set_options();
options.messages = false;
options.correct = false;
options.TTolerance = 1.0e-05;

amplitudes = linspace(1.0e-03, 0.3, 10);
XPredicted = zeros(658,length(amplitudes));
XCorrected = zeros(658,length(amplitudes));
for j=1:length(amplitudes)
  options.amplitude = amplitudes(j);
  [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, options);
  XPredicted(:,j) = x_pred;
  try
    XCorrected(:,j) = newtcorr(x_pred, v0);
  catch
    warning('Didn''t convergence to homoclinic solution')
  end
end

hold on
cm = lines;
plot3(XPredicted(homds.coords(1:homds.nphase:end),1:10), ...
      XPredicted(homds.coords(2:homds.nphase:end),1:10), ...
      XPredicted(homds.coords(3:homds.nphase:end),1:10), ...
      'color', cm(1,:), 'HandleVisibility', 'Off')
plot3(XCorrected(homds.coords(1:homds.nphase:end),1:10), ...
      XCorrected(homds.coords(2:homds.nphase:end),1:10), ...
      XCorrected(homds.coords(3:homds.nphase:end),1:10), ...
      '--', 'color', cm(2,:), 'HandleVisibility', 'Off')
plot3(bt1.x(1), bt1.x(2), bt1.x(3), '.', 'MarkerSize', 16)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(X,Y,Z)$ phase space')
xlabel('$X$')
ylabel('$Y$')
zlabel('$Z$')
grid on
view([71, 20]);
```



12.8 Convergence plot

We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

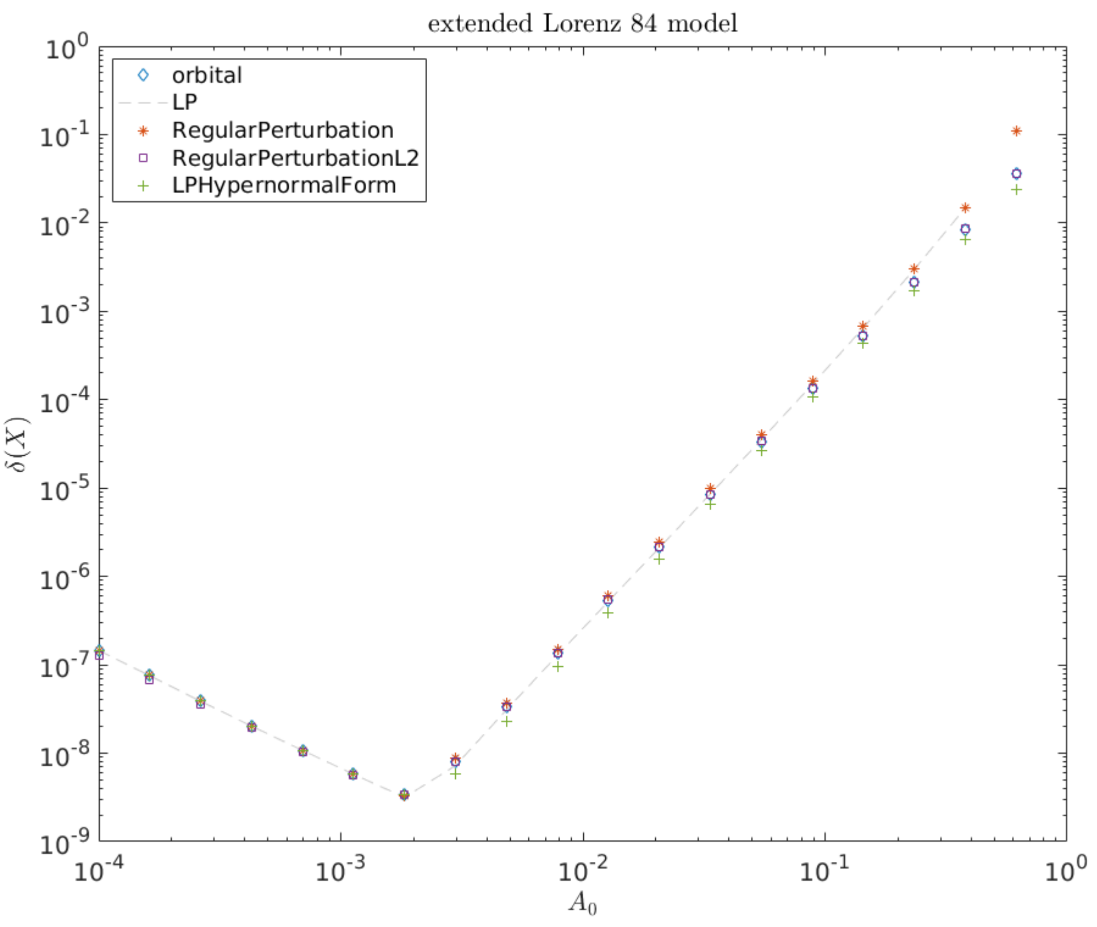
```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 0, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    BToptions.method = methodList{i};
    relativeErrors{i} = zeros(size(amplitudes));
    for j=1:length(amplitudes)
    BToptions.amplitude = amplitudes(j);
```

(continues on next page)

(continued from previous page)

```
    [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, BToptions);
    try
        x_corrected = newtcorr(x_pred, v0);
        relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
    catch
        warning('Did not converge.')
        continue
    end
  end
end

cm = lines();
loglog(amplitudes, relativeErrors{1}(:), 'd', ...
       amplitudes, relativeErrors{2}(:), '--', ...
       amplitudes, relativeErrors{3}(:), '*', ...
       amplitudes, relativeErrors{4}(:), 's', ...
       amplitudes, relativeErrors{5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('extended Lorenz 84 model')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```

```
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
```


extended Lorenz 84 model
orbital
LP
RegularPerturbation
RegularPerturbationL2
LPHypernormalForm
$\delta(X)$
A_0
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}

Part VII

SIR model

CHAPTER

THIRTEEN

GENERATE SYSTEM FILES FOR SIR MODEL WITH THE STANDARD INCIDENCE RATE

This Jupyter Notebook generates the **system file** for the following SIR model [SZ14] with the standard incidence rate

$$\begin{cases}\dfrac{dS}{dt} = A - dS - \dfrac{\beta SI}{S+I+R},\\ \dfrac{dI}{dt} = -(d+\nu)I - \mu(b,I)I + \dfrac{\beta SI}{S+I+R},\\ \dfrac{dR}{dt} = \mu(b,I)I - dR,\end{cases}$$

where $A > 0$ is the recruitment rate of susceptible population; $d > 0$ is the per capita natural death rate of the population; $\nu > 0$ is the per capita disease-induced death rate; $\mu > 0$ is the per capita recovery rate of infectious individual. The funtion μ is set to

$$\mu(b,I) = \mu_0 + (\mu_1 - \mu_0)\frac{b}{I+b}.$$

These are used in the *SIR model* demo.

13.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

13.2 Set the system name

```
system_name = 'SIRmodel';
```

13.3 Create coordinates and parameter names as strings

```
coordsnames = {'S', 'I', 'R'};
parnames = {'mu1', 'b'};
```

13.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `mu1` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});        % create symbol for alpha and delta
par=cell2sym(parnames);   % now alpha1 is par(1) etc
syms(coordsnames{:});     % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

13.5 Define fixed parameters

```
A = 20;
mu0 = 10;
d = 1/10;
nu = 1;
beta = 11.5;
```

13.6 Define the system

```
mu = @(b,I) mu0 + (mu1-mu0)*b/(I+b);
dS_dt  = A-d*S-beta*S*I/(S+I+R);
dI_dt  = -(d+nu)*I-mu(b,I)*I+beta*S*I/(S+I+R);
dR_dt  = mu(b,I)*I-d*R;
system = [dS_dt; dI_dt; dR_dt];
```

In general there are no modifications needed after this line.

13.7 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

13.8 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

FOURTEEN

SIR MODEL

In [SZ14] the following SIR model is considered

$$\begin{cases} \dfrac{dS}{dt} = A - dS - \dfrac{\beta SI}{S+I+R}, \\ \dfrac{dI}{dt} = -(d+\nu)I - \mu(b,I)I + \dfrac{\beta SI}{S+I+R}, \\ \dfrac{dR}{dt} = \mu(b,I)I - dR, \end{cases}$$

where $A > 0$ is the recruitment rate of susceptible population; $d > 0$ is the per capita natural death rate of the population; $\nu > 0$ is the per capita disease-induced death rate; $\mu > 0$ is the per capita recovery rate of infectious individual. The funtion μ is set to

$$\mu(b,I) = \mu_0 + (\mu_1 - \mu_0)\frac{b}{I+b}.$$

14.1 Overview

In this demo we will

- Compute a curve of equilibria, parametrized by b.
- Detect a Hopf and limit point.
- Start continuation form the Hopf point in two parameters (μ_1, b).
- Detect a Bogdanov-Takens points.
- Start continuation form the Bogdanov-Takens points in two parameters (μ_1, b).
- Compare the predicted and computed homoclinic bifurcation curve emanating from the Bogdanov-Takens point in parameters space.
- Compare a range of predictors for the homoclinic solutions emanating from the Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space.
- Create a convergence plot comparing the different homoclinic approximations derived in [Kuz21].

14.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Equilibrium'])
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
```

14.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *SIRmodel.ipynb*.

```
odefile=@SIRmodel;
```

14.4 Define equilibrium

As in [SZ14] we fix the parameters $A = 20, \mu_0 = 10, d = 1/10, \nu = 1$, and $\beta = 11.5$.

```
A     =  20;
mu0   =  10;
d     =  1/10;
nu    =  1;
beta  =  11.5;
```

For any endemic equilibrium $E(S, I, R)$, its coordinates satisfy

```
mu   =   @(b,mu1,I) mu0 + (mu1-mu0)*b/(I+b);
S    =   @(b,I,mu1) (A-(d+nu+mu(b, mu1, I))*I)/d;
R    =   @(b,I,mu1)   mu(b, mu1, I)*I/d;
```

and the coordinate I should be the positive root of the quadratic equation

$$f(I) = \mathcal{A}I^2 + \mathcal{B}I + \mathcal{C}$$

where

```
R0 = @(mu1) beta/(d+nu+mu1);
Ascr = (d+nu+mu0)*(beta-nu);
Bscr = @(b,mu1) (d+nu+mu0-beta)*A+(beta-nu)*(d+nu+mu1)*b;
Cscr = @(b,mu1) (d+nu+mu1)*A*b*(1-R0(mu1));
```

with

$$\mathbb{R}_0 = \frac{\beta}{d+\nu+\mu_1}$$

the basic reproduction number. The equation $f(I) = 0$ may have two roots if $\Delta_0 > 0$ which are denoted as

```
delta0 = d+nu+mu0;
delta1 = @(mu1) d+nu+mu1;
Delta0 = @(b,mu1) (beta-nu)^2*delta1(mu1)^2*b^2-2*A*(beta-nu)*(beta*(mu1-
 ↪mu0)+delta0*(delta1(mu1)-beta))*b+A^2*(beta-delta0)^2;
I1 = @(b,mu1)(-Bscr(b,mu1)-sqrt(Delta0(b,mu1)))/(2*Ascr);
I2 = @(b,mu1)(-Bscr(b,mu1)+sqrt(Delta0(b,mu1)))/(2*Ascr);
```

By setting $b = 0.022$ and $\mu_1 = 10.45$ we are slightly below the Hopf curve in Figure 4 of [SZ14].

```
b=0.022;
mu1=10.45;
p(1) = mu1;
p(2) = b;
x = [S(b,I2(b,mu1),mu1); I2(b,mu1); R(b,I2(b,mu1),mu1)];
```

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in *DDE-BifTool*.

```
parnames = {'mu1','b'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
ap = [ind.mu1, ind.b]; % continuation parameters
```

14.5 Continue equilibrium in parameter b

To continue the equilibrium in parameter b, we first need to obtain a tangent vector to the curve. This is done by the function `init_EP_EP`. Then we use the function `contset` to obtain a **struct** containing a list of options which is passed on to the continuer. By adjusting the values of the fields of the `opt` **struct** we set the minimum, initial and maximum set sizes, as well as the maximum number of points to continue and weather or not to detect bifurcation points (`opt.Singularities`) on the equilibrium curve. We also increase the accuracy for locating detected bifurcations (`TestTolerance`) and the maximum number of iterations that may be used to achieve this (`MaxTestIters`). This improves the homoclinic predictor which depend directly on the accuracy of the located Bogdanov-Takens point.

Finally, we continue the curve using the function `cont`.

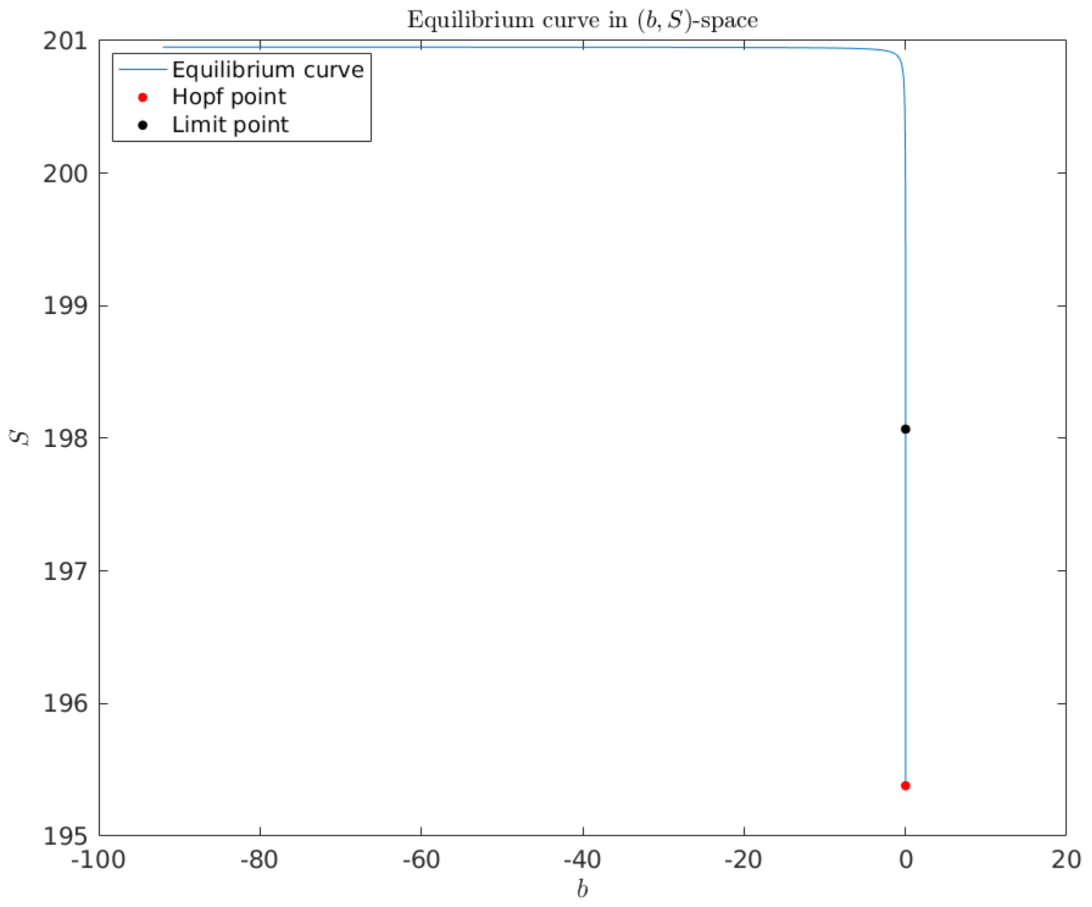
```
[x1_pred, v1_pred] = init_EP_EP(odefile, x, p, ind.b);
opt = contset;
opt.MaxNumPoints  = 1000;
opt.Singularities = 1;
opt.TestTolerance = 1e-15;
opt.MaxTestIters = 10;
opt.MinStepsize = 0.001;
[eqbr_x, ~, eqbr_bif_data] = cont(@equilibrium, x1_pred, v1_pred, opt);
```

```
first point found
tangent vector to first point found
label = H , x = ( 195.380284 0.041037 4.168310 0.022089 )
First Lyapunov coefficient = 3.908102e-04
label = LP, x = ( 198.069839 0.016933 1.743895 0.033419 )
a=-2.413445e+00

elapsed time  = 0.7 secs
npoints curve = 1000
```

There is one limit point detected (LP and one Hopf bifurcation point (H). The **array struct** `eqbr_bif_data` contains information about the detected bifurcation points. We use this to extract the index of the detected bifurcation points on the equilibrium curve `eqbr_x`. The equilibrium curve `eqbr_x` is just a two dimensional array. Each column consists of a point on the curve. The frist three rows contain the point (S, I, R) while the last row contains the parameter b.

Below we plot the equilibrium curve `eqbr_x`, together with the detect Hopf and limit point, in (b, S)-space.

```
%plot --width 1024 --height 800
hopfPointInfo   = eqbr_bif_data(strcmp({eqbr_bif_data.msg}, 'Hopf')==1);
limitPointsInfo = eqbr_bif_data(strcmp({eqbr_bif_data.msg}, 'Limit point')==1);
HopfPoint1 = eqbr_x(:,hopfPointInfo.index);
limitpoint1 = eqbr_x(:,limitPointsInfo.index);
plot(eqbr_x(4,:), eqbr_x(1,:)); hold on
plot(HopfPoint1(4), HopfPoint1(1), '.r' ,'MarkerSize', 20)
plot(limitpoint1(4), limitpoint1(1),  '.k' ,'MarkerSize', 20)
xlabel('$b$')
ylabel('$S$')
legend({'Equilibrium curve', 'Hopf point', 'Limit point'}, 'Location', 'NorthWest')
title('Equilibrium curve in $(b,S)$-space', 'fontsize', 15)
```



14.6 Continue Hopf point

We continue the Hopf point using again the function `cont`. We use the same continuation options as defined above in the **struct** `opt`.

```
hopf1.par = [mu1, b];
hopf1.par(ind.b) = eqbr_x(4, hopfPointInfo(1).index);
hopf1.x = eqbr_x(1:3, hopfPointInfo(1).index);
hopf1.x = x;
[hopf_x, hopf_v] = init_H_H(odefile, hopf1.x, hopf1.par', ap);
opt.MaxNumPoints = 500;
opt.MaxStepsize = .01;
opt.Singularities = 1;
[hopf_br, ~, hopf_br_bif] = cont(@hopf, hopf_x, hopf_v, opt);
```

```
first point found
tangent vector to first point found
label = GH, x = ( 195.599878 0.039046 3.970613 10.427886 0.025488 0.012691 )
l2=-8.624030e-05
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
↪RCOND =  1.423059e-17.
```

(continues on next page)

(continued from previous page)

```
> In nf_H (line 27)
In hopf>testf (line 150)
In cont>EvalTestFunc (line 810)
In cont (line 506)

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
↪RCOND =  4.847349e-18.
> In nf_H (line 28)
In hopf>testf (line 150)
In cont>EvalTestFunc (line 810)
In cont (line 506)

label = BT, x = ( 198.059382 0.017026 1.753334 10.450822 0.033239 0.000000 )
(a,b)=(-1.298117e-03, 1.219168e-02)

elapsed time  = 1.3 secs
npoints curve = 500
```

14.7 Continue homoclinic curve emanating from the Bogdanov-Takens point

To continue the homoclinic curve emanating from the Bogdanov-Takens point we use the functions `BT_Hom_set_options` and `init_BT_Hom` to obtain an initial approximation to the homoclinic solution (`hom_x`) as well as a tangent vector to the discretized homoclinic solution (`hom_v`) as described in *Initial prediction of homoclinic orbit near Bogdanov-Takens point 1*.

```
bt_points_info = hopf_br_bif(strcmp({hopf_br_bif.label}, 'BT')==1);
bt.x = hopf_br(1:3, bt_points_info.index);
bt.par = hopf_br(4:5, bt_points_info.index);
BToptions = BT_Hom_set_options();
[hom_x, hom_v] = init_BT_Hom(odefile, bt,  ap, BToptions);
opt = contset;
opt.Singularities = 0;
homoclinic_br = cont(@homoclinic, hom_x, hom_v, opt);
```

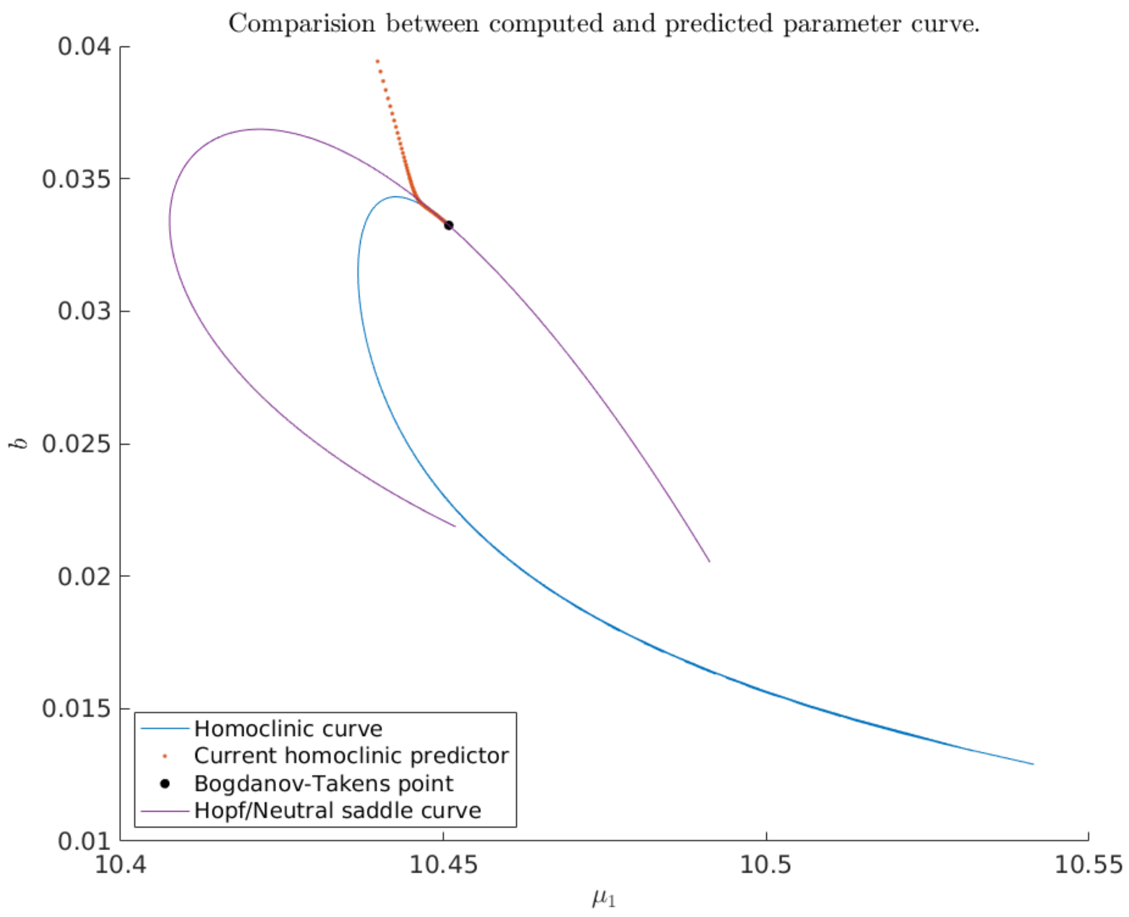
```
Center manifold coefficients' accuracy: 7.460699e-13
BT normal form coefficients:
a=-1.298117e-03,         b=1.219168e-02
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.524008
The initial half-return time T: 358.423
The initial distance eps0: 0.00239581
The initial distance eps1: 0.000941365
first point found
tangent vector to first point found

elapsed time  = 18.7 secs
npoints curve = 300
```

14.8 Compare predicted with computed parameters

Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm_orbital` to obtain the smooth orbital normal form coefficients, i.e. a and b, and the coefficients of the transformation K between the parameters of the system and the parameters of the smooth orbital normal form on the center manifold, see [Kuz21]. We also plot the Hopf curve which can be compared with Figure 4 in [SZ14].

```
global homds
hold on
% plot computed homoclinic parameter curve
plot(homoclinic_br(homds.PeriodIdx+1,:), ...
     homoclinic_br(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent smooth orbital normal form coefficients
bt = BT_nmfm_orbital(odefile, bt, ap, BToptions);
a   = bt.nmfm.a;
b   = bt.nmfm.b;
K10 = bt.nmfm.K10;
K01 = bt.nmfm.K01;
K02 = bt.nmfm.K02;
K11 = bt.nmfm.K11;
K03 = bt.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 0.3);
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alpha = bt.par(ap) + alpha;
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.')
% plot Bogdanov-Takens point
plot(bt.par(ind.mu1), bt.par(ind.b), '.k', 'MarkerSize', 20)
plot(hopf_br(4,:), hopf_br(5,:))
% set axis labels and legend
xlabel('$\mu_1$')
ylabel('$b$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point', 'Hopf/Neutral saddle curve'}, 'Location', 'SouthWest')
title('Comparision between computed and predicted parameter curve.')
```

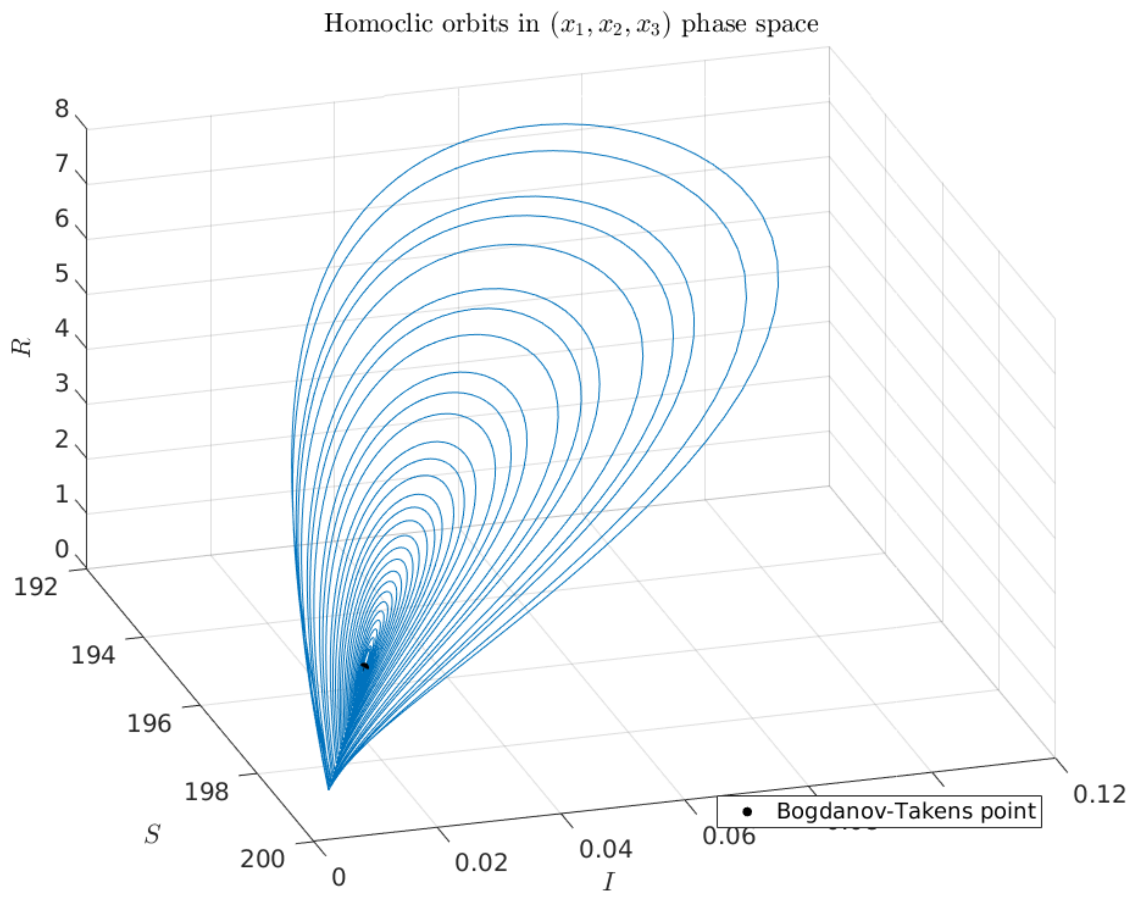
```
Center manifold coefficients' accuracy: 7.460699e-13
```



14.9 Plot of continued homoclinic solutions in phase-space

To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (S, I, R) phase-space.

```
hold on
plot3(homoclinic_br(homds.coords(1:homds.nphase:end), 1:10:end), ...
      homoclinic_br(homds.coords(2:homds.nphase:end), 1:10:end), ...
      homoclinic_br(homds.coords(3:homds.nphase:end), 1:10:end), ...
     'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$S$')
ylabel('$I$')
zlabel('$R$')
plot3(bt.x(1), bt.x(2), bt.x(3), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(x_1,x_2,x_3)$ phase space')
grid on
view([73, 33]);
```



14.9.1 Predictors of orbits for various epsilons

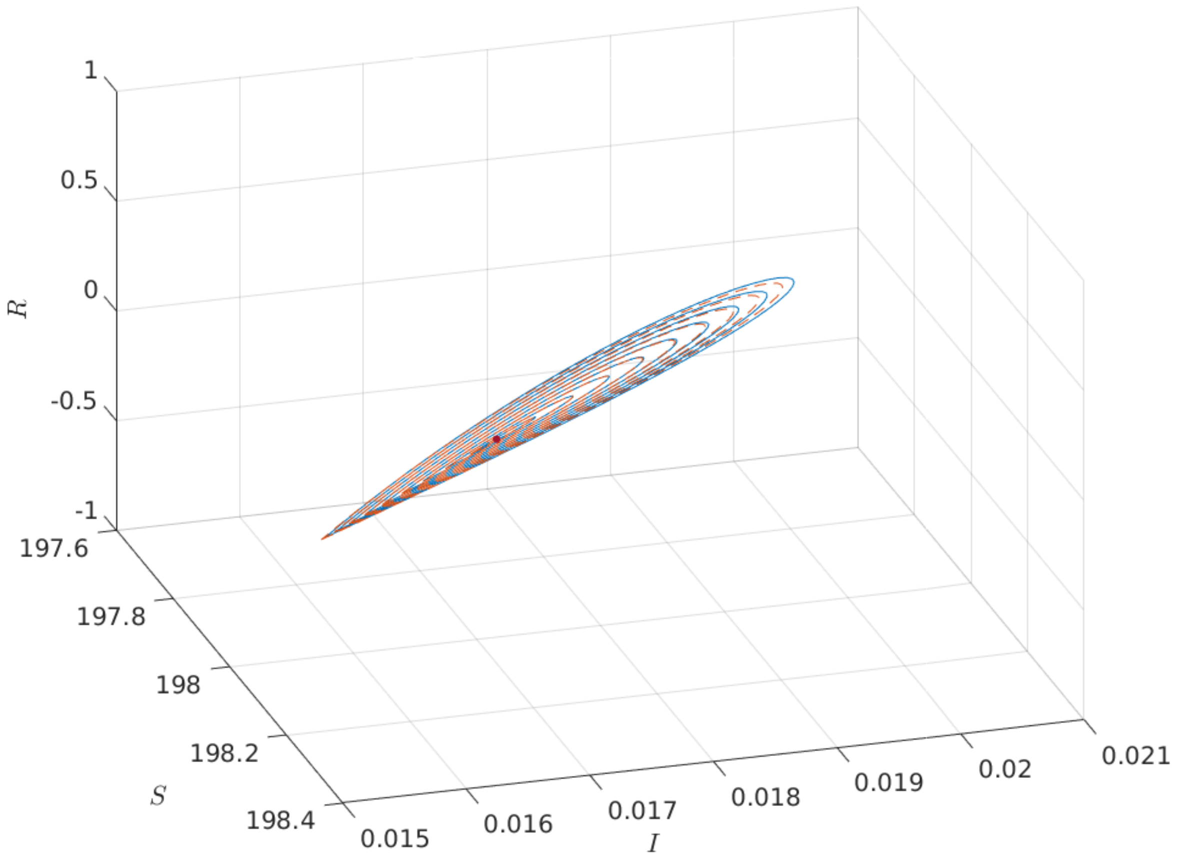
Below we compute for a large range of amplitudes the predicted and corrected homoclinic solutions and compare them in phase space. We see that even with an amplitude of `1` the predicted homoclinic orbit is still relatively close.

```
options = BT_Hom_set_options();
options.messages = false;
options.correct = false;
options.TTolerance = 1.0e-05;

amplitudes = linspace(1.0e-03, 1,10);
XPredicted = zeros(494,length(amplitudes));
XCorrected = zeros(494,length(amplitudes));
for j=1:length(amplitudes)
  options.amplitude = amplitudes(j);
  [x_pred, v0] = init_BT_Hom(odefile, bt, ap, options);
  XPredicted(:,j) = x_pred;
  try
    XCorrected(:,j) = newtcorr(x_pred, v0);
  catch
    warning('Didn''t convergence to homoclinic solution')
  end
```

(continues on next page)

(continued from previous page)

```
end

hold on
cm = lines;
plot(XPredicted(homds.coords(1:homds.nphase:end),1:10), ...
     XPredicted(homds.coords(2:homds.nphase:end),1:10), ...
      'color', cm(1,:))
plot(XCorrected(homds.coords(1:homds.nphase:end),1:10), ...
     XCorrected(homds.coords(2:homds.nphase:end),1:10), ...
      '--', 'color', cm(2,:))
plot(bt.x(1), bt.x(2), '.', 'MarkerSize', 16)
xlabel('$S$')
ylabel('$I$')
zlabel('$R$')
grid on
view([73, 33]);
```

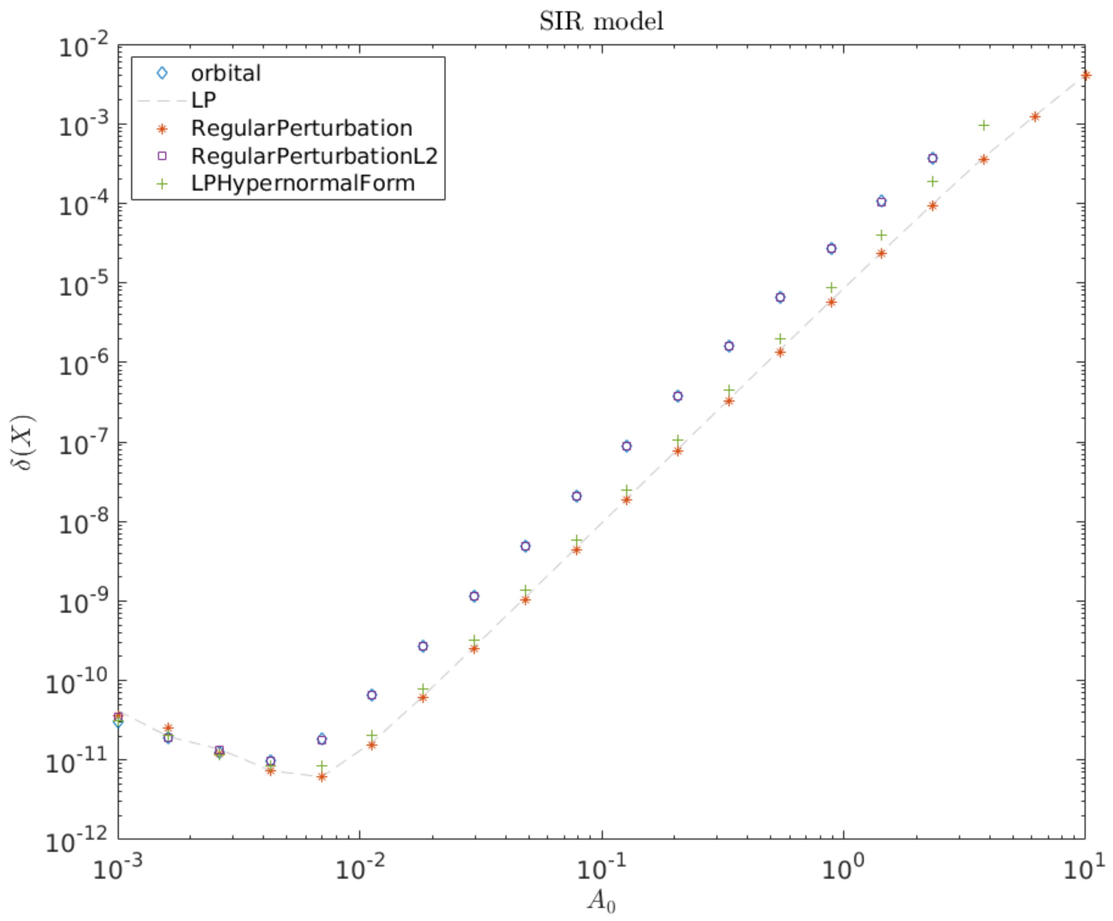


14.10 Convergence plot

We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-3, 1, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    BToptions.method = methodList{i};
    relativeErrors{i} = zeros(size(amplitudes));
    for j=1:length(amplitudes)
    BToptions.amplitude = amplitudes(j);
    [x_pred, v0] = init_BT_Hom(odefile, bt, ap, BToptions);
    try
        x_corrected = newtcorr(x_pred, v0);
        relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
    catch
        warning('Did not converge.')
        continue
    end
  end
end

cm = lines();
loglog(amplitudes, relativeErrors{1}(:), 'd', ...
       amplitudes, relativeErrors{2}(:), '--', ...
       amplitudes, relativeErrors{3}(:), '*', ...
       amplitudes, relativeErrors{4}(:), 's', ...
       amplitudes, relativeErrors{5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('SIR model')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```

```
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
```


SIR model
orbital
LP
RegularPerturbation
RegularPerturbationL2
LPHypernormalForm
$\delta(X)$
A_0
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-11}
10^{-12}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}

Part VIII

Hodgkin-Huxley

CHAPTER

FIFTEEN

GENERATE SYSTEM FILES FOR HODGKIN-HUXLEY EQUATIONS

The Hodgkin-Huxley equations [HH52a] relate the difference in electric potential across the cell membrane (V) and gating variables $(m, n$ and h) for ion channels to the stimulus intensity (I) and temperature (T), as follows:

$$\begin{cases} \dot{V} = & -G(V, m, n, h) + I \\ \dot{m} = & \Phi(T)\left[(1-m)\alpha_m(V) - m\beta_m(V)\right] \\ \dot{n} = & \Phi(T)\left[(1-n)\alpha_n(V) - n\beta_n(V)\right] \\ \dot{h} = & \Phi(T)\left[(1-h)\alpha_h(V) - h\beta_h(V)\right] \end{cases}$$

where $\dot{x}$ stands for $\mathrm{d}x/\mathrm{d}t$ and Φ is given by $\Phi(T) = 3^{(\mathrm{T}-6.3)/10}$. The other functions involved are:

$$G(V, m, n, h) = \bar{g}_{\mathrm{Na}} m^3 h\left(V - \bar{V}_{\mathrm{Na}}\right) + \bar{g}_{\mathrm{K}} n^4\left(V - \bar{V}_{\mathrm{K}}\right) + \bar{g}_{\mathrm{L}}\left(V - \bar{V}_{\mathrm{L}}\right)$$

and the equations modeling the variation of membrane permeability are:

$$\begin{array}{ll} \alpha_m(V) = \Psi\left(\frac{V+25}{10}\right) & \beta_m(V) = 4e^{V/18} \\ \alpha_n(V) = 0.1\Psi\left(\frac{V+10}{10}\right) & \beta_n(V) = 0.125e^{V/80} \\ \alpha_h(V) = 0.07e^{V/20} & \beta_h(V) = \left(1 + e^{(V+30)/10}\right)^{-1} \end{array}$$

with

$$\Psi(x) = \begin{cases} x/\left(e^x - 1\right) & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases}$$

The parameters $\bar{g}_{\text{ion}}$ and $\bar{V}_{\text{ion}}$ representing maximum conductance and equilibrium potential for the ion were obtained from experimental data by Hodgkin and Huxley, with the values given below:

$$\begin{array}{lll} \bar{g}_{\mathrm{Na}} = 120\mathrm{mS/cm}^2, & \bar{g}_{\mathrm{K}} = 36\mathrm{mS/cm}^2, & \bar{g}_{\mathrm{L}} = 0.3\mathrm{mS/cm}^2 \\ \bar{V}_{\mathrm{Na}} = -115\mathrm{mV}, & \bar{V}_{\mathrm{K}} = 12\mathrm{mV}, & \bar{V}_{\mathrm{L}} = 10.599\mathrm{mV} \end{array}$$

The values of $\bar{V}_{\mathrm{Na}}$ and $\bar{V}_{\mathrm{K}}$ can be controlled experimentally [HH52b]. The temperature is set to $T = 6.3°$.

These are used in the *HodgkinHuxley*.

15.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

15.2 Set the system name

```
system_name = 'HodgkinHuxley';
```

15.3 Create coordinates and parameter names as strings

```
coordsnames = {'V', 'm', 'n', 'h'};
parnames={'VbarK', 'I'};
```

15.4 Create symbols for parameters

The array `|par|` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `mu_1` or `par(1)`

15.5 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `k` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});       % create symbol for alpha and delta
par=cell2sym(parnames);  % now alpha1 is par(1) etc
syms(coordsnames{:});    % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

15.6 Define fixed parameters

```
gbarNa = 120;
gbarK  = 36;
gbarL  = 0.3;
VbarNa = -115;
VbarL  = 10.599;
T = 6.3;
```

15.7 Define the system

```
Psi = @(x) x/(exp(x)-1);
alpha_m = @(V) Psi( (V+25)/10 );
alpha_n = @(V) 0.1*Psi( (V+10)/10);
alpha_h = @(V) 0.07*exp(V/20);

beta_m = @(V) 4*exp(V/18);
beta_n = @(V) 0.125*exp(V/80);
beta_h = @(V) 1/(1+exp((V+30)/10));
```

(continues on next page)

(continued from previous page)

```
G = @(V, m, n, h) gbarNa*m^3*h*(V-VbarNa) + gbarK*n^4*(V-VbarK) + gbarL*(V-VbarL);
Phi = @(T) 3^(T-6.3)/10;
dV_dt = -G(V, m, n, h)+I;
dm_dt = Phi(T)*((1-m)*alpha_m(V)-m*beta_m(V));
dn_dt = Phi(T)*((1-n)*alpha_n(V)-n*beta_n(V));
dh_dt = Phi(T)*((1-h)*alpha_h(V)-h*beta_h(V));
system = [dV_dt; dm_dt; dn_dt; dh_dt];
```

In general there are no modifications needed after this line.

15.8 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

15.9 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

SIXTEEN

HODGKIN HUXLEY EQUATIONS

The Hodgkin-Huxley equations [HH52a] relate the difference in electric potential across the cell membrane (V) and gating variables $(m, n$ and h) for ion channels to the stimulus intensity (I) and temperature (T), as follows:

$$\begin{cases} \dot{V} = -G(V,m,n,h) + I, \\ \dot{m} = \Phi(T)\left[(1-m)\alpha_m(V) - m\beta_m(V)\right], \\ \dot{n} = \Phi(T)\left[(1-n)\alpha_n(V) - n\beta_n(V)\right], \\ \dot{h} = \Phi(T)\left[(1-h)\alpha_h(V) - h\beta_h(V)\right], \end{cases} \tag{1}$$

where $\dot{x}$ stands for $\mathrm{d}x/\mathrm{d}t$ and Φ is given by $\Phi(T) = 3^{(\mathrm{T}-6.3)/10}$. The other functions involved are:

$$G(V,m,n,h) = \bar{g}_{\mathrm{Na}} m^3 h \left(V - \bar{V}_{\mathrm{Na}}\right) + \bar{g}_{\mathrm{K}} n^4 \left(V - \bar{V}_{\mathrm{K}}\right) + \bar{g}_{\mathrm{L}} \left(V - \bar{V}_{\mathrm{L}}\right)$$

and the equations modeling the variation of membrane permeability are:

$$\begin{array}{ll} \alpha_m(V) = \Psi\left(\frac{V+25}{10}\right), & \beta_m(V) = 4e^{V/18}, \\ \alpha_n(V) = 0.1\Psi\left(\frac{V+10}{10}\right), & \beta_n(V) = 0.125e^{V/80} \\ \alpha_h(V) = 0.07e^{V/20}, & \beta_h(V) = \left(1 + e^{(V+30)/10}\right)^{-1}, \end{array}$$

with

$$\Psi(x) = \begin{cases} x/\left(e^x - 1\right), & \text{if } x \neq 0, \\ 1, & \text{if } x = 0. \end{cases}$$

The parameters $\bar{g}_{\text{ion}}$ and $\bar{V}_{\text{ion}}$ representing maximum conductance and equilibrium potential for the ion were obtained from experimental data by Hodgkin and Huxley, with the values given below:

$$\begin{array}{lll} \bar{g}_{\mathrm{Na}} = 120\mathrm{mS/cm}^2, & \bar{g}_{\mathrm{K}} = 36\mathrm{mS/cm}^2, & \bar{g}_{\mathrm{L}} = 0.3\mathrm{mS/cm}^2, \\ \bar{V}_{\mathrm{Na}} = -115\mathrm{mV}, & \bar{V}_{\mathrm{K}} = 12\mathrm{mV}, & \bar{V}_{\mathrm{L}} = 10.599\mathrm{mV}. \end{array}$$

The values of $\bar{V}_{\mathrm{Na}}$ and $\bar{V}_{\mathrm{K}}$ can be controlled experimentally [HH52b]. The temperature is set to $T = 6.3^\circ$.

16.1 Overview

In this demo we will

- Manually define an Bogdanov-Takens point.
- Start continuation of the homoclinic branch emanating from the Bogdanov-Takens points in two parameters $(\bar{V}_{\mathrm{K}}, I)$ using the new homoclinic smooth orbital predictor from [Kuz21].
- Compare the predicted and computed homoclinic bifurcation curve emanating from the Bogdanov-Takens point in parameters space.

- Compare a range of predictors for the homoclinic solutions emanating from the Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space.
- Create a convergence plot of the different homoclinic approximations derived in [Kuz21].

16.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Equilibrium'])
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
```

16.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *Generate system files for Hodgkin-Huxley equations*.

```
odefile=@HodgkinHuxley;
```

16.4 Define Bogdanov-Takens point manually

By solving for real equilibria of (1) with double zero eigenvalues we obtain two equilibria. The first equilibrium is given by

$$(V, m, n, h) = (-2.835463618170097, 0.07351498630356315, 0.361877602925177, 0.494859128785482),$$

with parameter values $V_{\bar{K}} = (-4.977020454108788, -0.06185214966177632)$. The second solution we can discard, since the geometric multiplicity is equation to two.

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in *DDE-BifTool*.

```
parnames = {'VbarK', 'I'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
ap = [ind.VbarK ind.I]; % continuation parameters
```

Next we define the Bogdanov-Takens point

```
V = -2.835463618170097;
m = 0.07351498630356315;
n = 0.361877602925177;
h = 0.494859128785482;
VbarK = -4.977020454108788;
I = -0.06185214966177632;
bt1.x = [V; m; n; h];
bt1.par = [VbarK; I];
```

16.5 Continue homoclinic curve emanating from the Bogdanov-Takens point

To continue the homoclinic curve emanating from the Bogdanov-Takens point we use the functions `BT_Hom_set_options` and `init_BT_Hom` to obtain an initial approximation to the homoclinic solution (`hom_x`) as well as a tangent vector to the discretized homoclinic solution (`hom_v`) as described in *Initial prediction of homoclinic orbit near Bogdanov-Takens point 1*.

```
ap = [ind.VbarK, ind.I];
BToptions = BT_Hom_set_options();
BToptions.ntst = 40;
[hom_x, hom_v] = init_BT_Hom(odefile, bt1,  ap, BToptions);
opt = contset;
opt.MaxNumPoints = 1000;
opt.Singularities = 0;
homoclinic_br1 = cont(@homoclinic, hom_x, hom_v, opt);
```

```
Center manifold coefficients' accuracy: 1.705303e-13
BT normal form coefficients:
a=2.551541e-05,         b=-7.458949e-03
The initial perturbation parameter epsilon:  1.000000e-01
The initial amplitude: 0.0275168
The initial half-return time T: 12114.7
The initial distance eps0: 4.84346e-05
The initial distance eps1: 1.48445e-05
first point found
tangent vector to first point found

elapsed time  = 66.8 secs
npoints curve = 1000
```

16.6 Compare predicted with computed parameters

Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm_orbital` to obtain the smooth orbital normal form coefficients, i.e. a and b, and the coefficients of the transformation K between the parameters of the system and the parameters of the smooth orbital normal form on the center manifold, see [Kuz21].

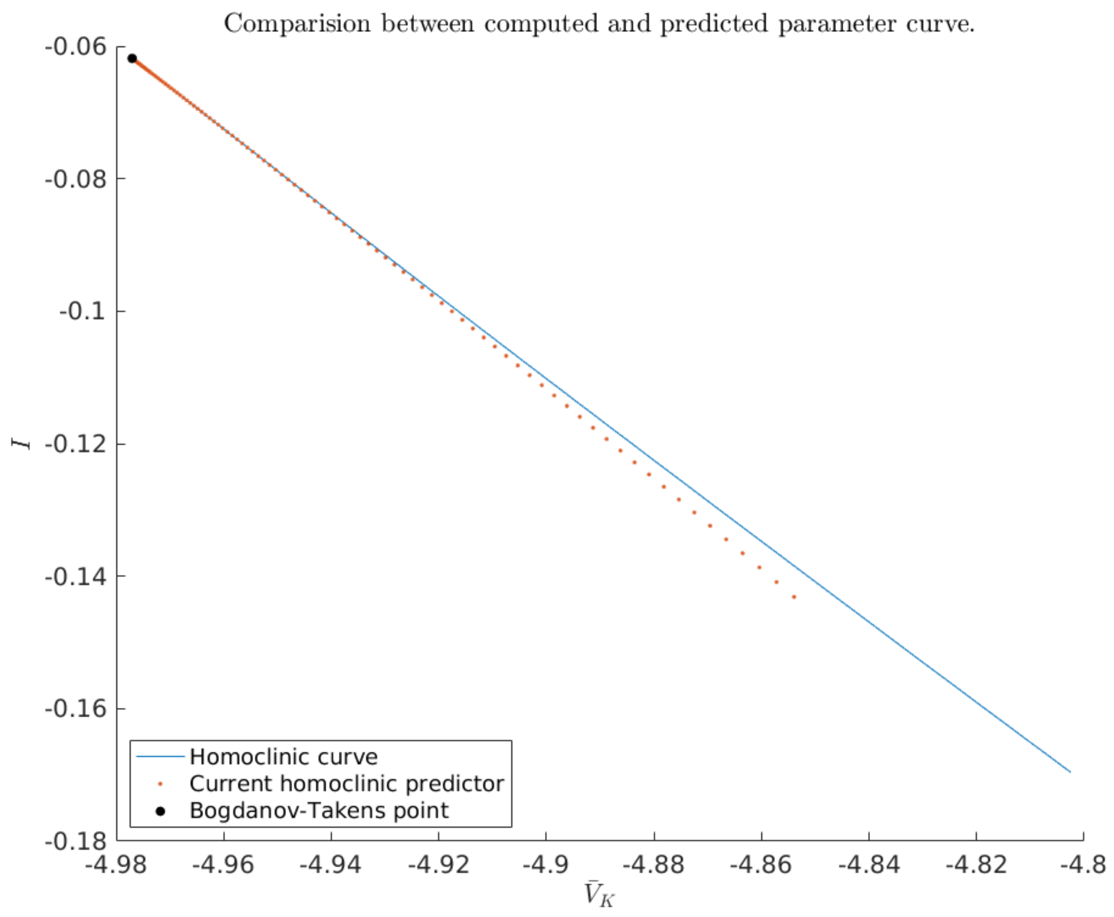
```
%plot --width 1024 --height 800
hold on
```

(continues on next page)

(continued from previous page)

```
global homds
% plot computed parameter curve
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent normal form coefficients
bt1 = BT_nmfm_orbital(odefile, bt1, ap, BToptions);
a   = bt1.nmfm.a;
b   = bt1.nmfm.b;
K10 = bt1.nmfm.K10;
K01 = bt1.nmfm.K01;
K02 = bt1.nmfm.K02;
K11 = bt1.nmfm.K11;
K03 = bt1.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 0.8);
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alpha = bt1.par(ap) + alpha;
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.')
% plot Bogdanov-Takens point
plot(bt1.par(ind.VbarK), bt1.par(ind.I), '.k', 'MarkerSize', 20)
% set axis labels and legend
xlabel('$\bar{V}_K$')
ylabel('$I$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'SouthWest')
title('Comparision between computed and predicted parameter curve.')
```

```
Center manifold coefficients' accuracy: 1.705303e-13
```

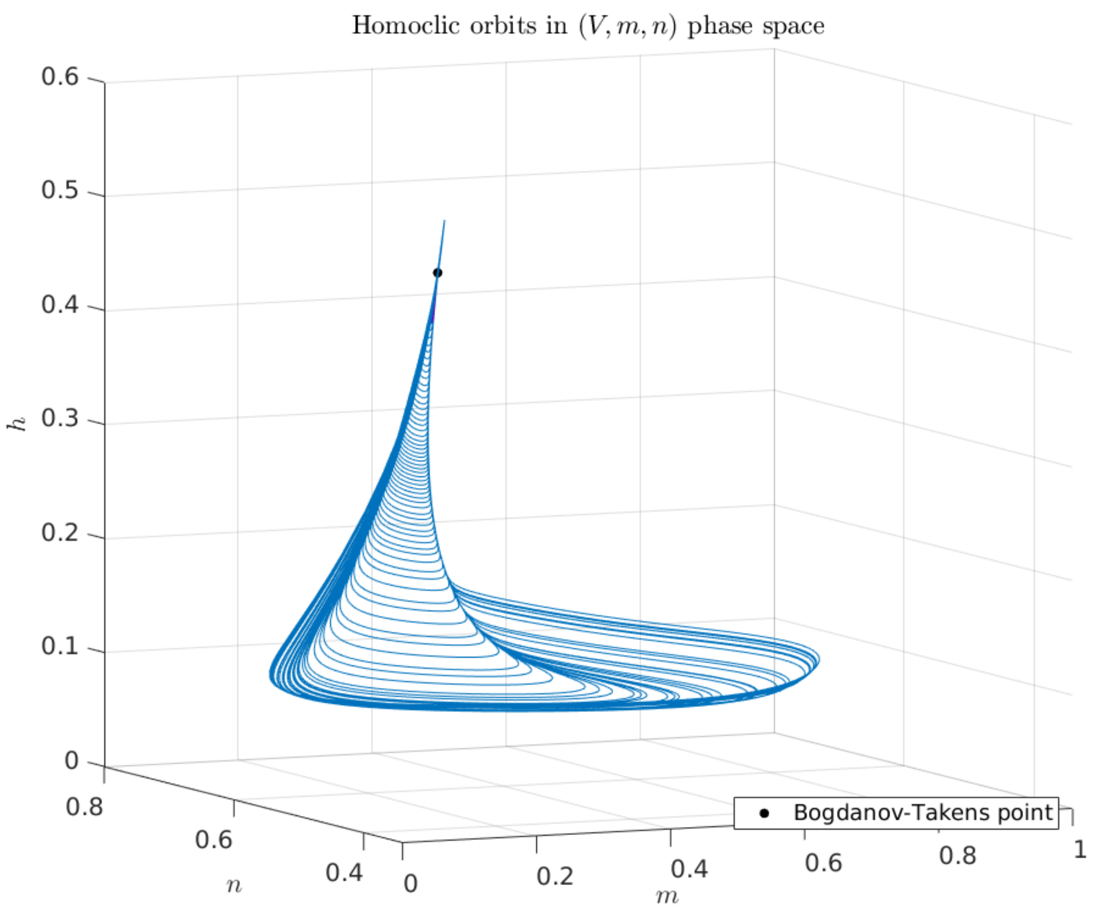


16.7 Plot of continued homoclinic solutions

To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (m, n, h) phase-space.

```
%plot native
hold on
plot3(homoclinic_br1(homds.coords(2:homds.nphase:end), 1:60), ...
      homoclinic_br1(homds.coords(3:homds.nphase:end), 1:60), ...
      homoclinic_br1(homds.coords(4:homds.nphase:end), 1:60), ...
      'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
plot3(homoclinic_br1(homds.coords(2:homds.nphase:end), 61:100), ...
      homoclinic_br1(homds.coords(3:homds.nphase:end), 61:100), ...
      homoclinic_br1(homds.coords(4:homds.nphase:end), 61:100), ...
      'Color', [0.4470 0 0.7410], 'HandleVisibility', 'Off')
xlabel('$m$')
ylabel('$n$')
zlabel('$h$')
plot3(bt1.x(2), bt1.x(3), bt1.x(4), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(V,m,n)$ phase space')
grid on
view([-24, 7]);
```

```
%plot inline
hold on
plot3(homoclinic_br1(homds.coords(2:homds.nphase:end), 1:10:end), ...
      homoclinic_br1(homds.coords(3:homds.nphase:end), 1:10:end), ...
      homoclinic_br1(homds.coords(4:homds.nphase:end), 1:10:end), ...
      'Color', [0 0.4470 0.7410], 'HandleVisibility', 'Off')
xlabel('$m$')
ylabel('$n$')
zlabel('$h$')
plot3(bt1.x(2), bt1.x(3), bt1.x(4), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(V,m,n)$ phase space')
grid on
view([-24, 7]);
```



16.7.1 Predictors of orbits for various epsilons

Below we compute for a range of amplitudes the predicted and corrected homoclinic solutions and compare them in phase space. As seen from the plot *above* the homoclinic solutions start to deform rather quickly. Therefore, we cannot expect the approximation to be accurate for relative large amplitude values. Still, for an amplitude of 0.3 the approximation is nearby the corrected homoclinic solutions.

```
%plot native
options = BT_Hom_set_options();
options.messages = false;
options.correct = false;
options.TTolerance = 1.0e-05;

amplitudes = linspace(1.0e-02, 0.3, 5);
XPredicted = zeros(658,length(amplitudes));
XCorrected = zeros(658,length(amplitudes));
for j=1:length(amplitudes)
  options.amplitude = amplitudes(j);
  [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, options);
  XPredicted(:,j) = x_pred;
  try
    XCorrected(:,j) = newtcorr(x_pred, v0);
  catch
    warning('Didn''t convergence to homoclinic solution')
  end
end

hold on
cm = lines;
plot3(XPredicted(homds.coords(2:homds.nphase:end),1:end), ...
      XPredicted(homds.coords(3:homds.nphase:end),1:end), ...
      XPredicted(homds.coords(4:homds.nphase:end),1:end), ...
      'color', cm(1,:), 'HandleVisibility', 'Off')
plot3(XCorrected(homds.coords(2:homds.nphase:end),1:end), ...
      XCorrected(homds.coords(3:homds.nphase:end),1:end), ...
      XCorrected(homds.coords(4:homds.nphase:end),1:end), ...
      '--', 'color', cm(2,:), 'HandleVisibility', 'Off')
plot3(bt1.x(2), bt1.x(3), bt1.x(4), '.', 'MarkerSize', 16)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(X,Y,Z)$ phase space')
xlabel('$m$')
ylabel('$n$')
zlabel('$h$')
grid on
view([-24, 7]);
```

16.8 Convergence plot

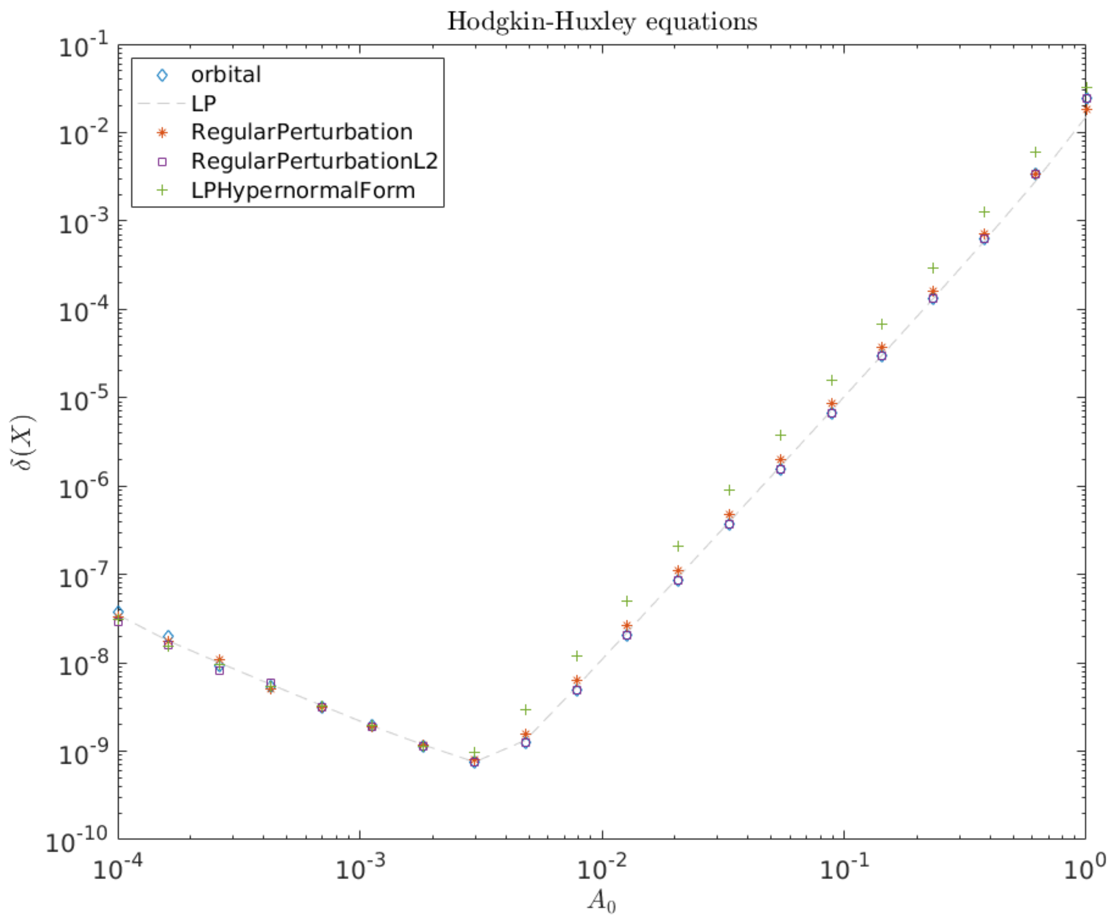
We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
```

(continues on next page)

(continued from previous page)

```
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 0, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    for o=1:3
        BToptions.method = methodList{i};
        BToptions.order = o;
        relativeErrors{o,i} = zeros(size(amplitudes));
        for j=1:length(amplitudes)
            BToptions.amplitude = amplitudes(j);
            [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, BToptions);
            try
                x_corrected = newtcorr(x_pred, v0);
                relativeErrors{o,i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
            catch
                warning('Did not converge.')
                continue
            end
        end
    end
end

cm = lines();
loglog(amplitudes, relativeErrors{3,1}(:), 'd', ...
       amplitudes, relativeErrors{3,2}(:), '--', ...
       amplitudes, relativeErrors{3,3}(:), '*', ...
       amplitudes, relativeErrors{3,4}(:), 's', ...
       amplitudes, relativeErrors{3,5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('Hodgkin-Huxley equations')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```


Hodgkin-Huxley equations
orbital
LP
RegularPerturbation
RegularPerturbationL2
LPHypernormalForm
$\delta(X)$
A_0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
10^{-10}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}

CHAPTER

SEVENTEEN

SAVE DATA TO FILES

```
writematrix([amplitudes', relativeErrors{1,3}(:)], '../../data/HHRPorder1.csv',
 ↪'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{2,3}(:)], '../../data/HHRPorder2.csv',
 ↪'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{3,3}(:)], '../../data/HHRPorder3.csv',
 ↪'Delimiter', ' ')

writematrix([amplitudes', relativeErrors{1,2}(:)], '../../data/HHLPorder1.csv',
 ↪'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{2,2}(:)], '../../data/HHLPorder2.csv',
 ↪'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{3,2}(:)], '../../data/HHLPorder3.csv',
 ↪'Delimiter', ' ')

writematrix([amplitudes', relativeErrors{3,1}(:)], '../../data/HHLPorder3orbital.csv',
 ↪ 'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{3,4}(:)], '../../data/
 ↪HHRegularPerturbationL2.csv', 'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{3,5}(:)], '../../data/HHLPHypernormalForm.csv
 ↪', 'Delimiter', ' ')
```

Part IX

Homoclinic RG flows

CHAPTER

EIGHTEEN

GENERATE SYSTEM FILES FOR HOMOCLINIC RG FLOWS

This Jupyter Notebook generates the **system files** for the model considered in the *HomoclinicRGflows* demo.

18.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

18.2 Set the system name

```
system_name = 'HomoclinicRGflows';
```

18.3 Create coordinates and parameter names as strings

```
coordsnames = {'g1', 'g2', 'g3', 'g4'};
parnames={'epsilon','M', 'N'};
```

18.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `epsilon` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});        % create symbol for alpha and delta
par=cell2sym(parnames);   % now alpha1 is par(1) etc
syms(coordsnames{:});     % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

18.5 Define beta functions

```
beta12 = @(M,N) (1/8).*N.^(-2).*(96.*g2.^2.*g4.*((-8)+N).*N+768.*g2.*g3.*g4.*N.^2+ ...
128.*g1.*g3.*g4.*N.^2.*(3+5.*M+N+N.^2)+16.*g2.*g3.^2.*N.^2.*(16+ ...
7.*M+N+N.^2)+32.*g1.^2.*g4.*N.*((-40)+(-8).*M+8.*N+7.*M.*N+8.* ...
N.^2)+64.*g1.*g4.^2.*N.^2.*(22+(-2).*M+M.*N+M.*N.^2)+16.*g1.* ...
g3.^2.*N.^2.*(32+2.*M+N+M.*N+N.^2+M.*N.^2)+64.*g1.*g2.*g4.*N.*(( ...
-32)+(-4).*M+16.*N+2.*M.*N+5.*N.^2+2.*M.*N.^2)+4.*g2.^2.*g3.*N.*(( ...
-256)+(-64).*M+72.*N+10.*M.*N+19.*N.^2+2.*M.*N.^2)+16.*g1.^2.*g3.* ...
N.*((-80)+(-24).*M+30.*N+4.*M.*N+7.*N.^2+4.*M.*N.^2)+16.*g1.*g2.* ...
g3.*N.*((-144)+(-40).*M+42.*N+17.*M.*N+21.*N.^2+5.*M.*N.^2)+ ...
g2.^3.*(896+128.*M+(-352).*N+(-48).*M.*N+(-12).*N.^2+(-12).*M.* ...
N.^2+7.*N.^3+2.*M.*N.^3)+4.*g1.^2.*g2.*(928+224.*M+(-392).*N+( ...
-100).*M.*N+(-2).*N.^2+11.*M.*N.^2+23.*N.^3+7.*M.*N.^3+4.*N.^4)+ ...
4.*g1.^3.*(352+96.*M+(-152).*N+(-44).*M.*N+10.*N.^3+3.*M.*N.^3+M.* ...
N.^4)+2.*g1.*g2.^2.*(1600+320.*M+(-656).*N+(-136).*M.*N+32.*N.^2+ ...
10.*M.*N.^2+50.*N.^3+10.*M.*N.^3+5.*N.^4+2.*M.*N.^4)).*pi.^(-2);

beta22 = @(M,N) (1/8).*N.^(-2).*(1536.*g1.*g3.*g4.*N.^2+128.*g2.*g3.*g4.*N.^2.*(9+ ...
5.*M+N+N.^2)+32.*g1.*g3.^2.*N.^2.*(16+7.*M+N+N.^2)+192.*g1.^2.* ...
g4.*N.*((-12)+(-2).*M+4.*N+N.^2)+64.*g1.*g2.*g4.*N.*((-80)+(-16).* ...
M+16.*N+5.*M.*N+7.*N.^2)+64.*g2.*g4.^2.*N.^2.*(22+(-2).*M+M.*N+M.* ...
N.^2)+16.*g2.*g3.^2.*N.^2.*(48+9.*M+2.*N+M.*N+2.*N.^2+M.*N.^2)+ ...
16.*g1.^2.*g3.*N.*((-128)+(-32).*M+32.*N+12.*M.*N+14.*N.^2+3.*M.* ...
N.^2)+16.*g1.*g2.*g3.*N.*((-272)+(-72).*M+82.*N+15.*M.*N+24.*N.^2+ ...
6.*M.*N.^2)+16.*g2.^2.*g4.*N.*((-176)+(-40).*M+58.*N+14.*M.*N+17.* ...
N.^2+11.*M.*N.^2)+4.*g2.^2.*g3.*N.*((-576)+(-160).*M+176.*N+54.* ...
M.*N+74.*N.^2+17.*M.*N.^2)+8.*g1.^3.*(288+64.*M+(-112).*N+(-24).* ...
M.*N+(-6).*N.^2+3.*M.*N.^2+5.*N.^3+2.*M.*N.^3+N.^4)+2.*g1.*g2.^2.* ...
(3968+1024.*M+(-1600).*N+(-416).*M.*N+(-56).*N.^2+(-22).*M.*N.^2+ ...
85.*N.^3+17.*M.*N.^3+11.*N.^4)+4.*g1.^2.*g2.*(1856+448.*M+(-736).* ...
N+(-176).*M.*N+(-22).*N.^2+(-5).*M.*N.^2+43.*N.^3+8.*M.*N.^3+3.* ...
N.^4+2.*M.*N.^4)+g2.^3.*(2816+768.*M+(-1152).*N+(-320).*M.*N+12.* ...
N.^2+(-12).*M.*N.^2+69.*N.^3+30.*M.*N.^3+7.*N.^4+5.*M.*N.^4)).* ...
pi.^(-2);

beta32 = @(M,N) (1/8).*N.^(-3).*(384.*g1.*g2.*g4.*N.*(4+N.^2)+96.*g2.^2.*g4.*N.*( ...
8+N.^2)+128.*g1.*g3.*g4.*N.^2.*((-10)+(-2).*M+5.*N+M.*N+5.*N.^2)+ ...
32.*g3.^2.*g4.*N.^3.*(18+14.*M+7.*N+7.*N.^2)+64.*g3.*g4.^2.*N.^3.* ...
(22+(-2).*M+M.*N+M.*N.^2)+96.*g1.^2.*g4.*N.*(8+2.*N.^2+M.*N.^2)+ ...
24.*g3.^3.*N.^3.*(16+2.*M+2.*N+M.*N+2.*N.^2+M.*N.^2)+64.*g2.*g3.* ...
g4.*N.^2.*((-20)+(-4).*M+10.*N+2.*M.*N+5.*N.^2+2.*M.*N.^2)+16.* ...
g1.*g3.^2.*N.^2.*((-52)+(-14).*M+26.*N+7.*M.*N+14.*N.^2+7.*M.* ...
N.^2)+8.*g2.*g3.^2.*N.^2.*((-104)+(-28).*M+52.*N+14.*M.*N+38.* ...
N.^2+7.*M.*N.^2)+16.*g1.*g2.*g3.*N.*(208+56.*M+(-48).*N+(-12).*M.* ...
N+12.*N.^2+3.*M.*N.^2+8.*N.^3+M.*N.^3+2.*N.^4)+8.*g1.^2.*g3.*N.*( ...
208+56.*M+(-48).*N+(-12).*M.*N+12.*N.^2+6.*M.*N.^2+4.*N.^3+3.*M.* ...
N.^3+M.*N.^4)+8.*g1.^3.*((-96)+(-16).*M+24.*N+4.*M.*N+(-20).*N.^2+ ...
(-10).*M.*N.^2+6.*N.^3+2.*M.*N.^3+N.^4+M.*N.^4)+4.*g2.^2.*g3.*N.*( ...
416+112.*M+(-96).*N+(-24).*M.*N+18.*N.^2+6.*M.*N.^2+12.*N.^3+4.* ...
M.*N.^3+2.*N.^4+M.*N.^4)+g2.^3.*((-768)+(-128).*M+192.*N+32.*M.*N+ ...
(-96).*N.^2+32.*N.^3+4.*M.*N.^3+7.*N.^4+2.*M.*N.^4)+2.*g1.*g2.^2.* ...
((-1152)+(-192).*M+288.*N+48.*M.*N+(-176).*N.^2+(-40).*M.*N.^2+ ...
70.*N.^3+10.*M.*N.^3+21.*N.^4+2.*M.*N.^4)+4.*g1.^2.*g2.*((-576)+( ...
-96).*M+144.*N+24.*M.*N+(-104).*N.^2+(-40).*M.*N.^2+34.*N.^3+13.* ...
M.*N.^3+11.*N.^4+3.*M.*N.^4)).*pi.^(-2);
```

(continues on next page)

(continued from previous page)

```
beta42 = @(M,N) (1/8).*N.^(-3).*(224.*g3.*g4.^2.*N.^3.*(2.*M+N+N.^2)+24.*g3.^3.* ...
N.^3.*(4+2.*M+N+N.^2)+224.*g1.*g4.^2.*N.^2.*((-4)+(-2).*M+2.*N+M.* ...
N+2.*N.^2)+16.*g1.*g3.^2.*N.^2.*((-16)+(-2).*M+8.*N+M.*N+7.*N.^2)+ ...
96.*g4.^3.*N.^3.*(8+(-2).*M+M.*N+M.*N.^2)+32.*g3.^2.*g4.*N.^3.*( ...
22+N+M.*N+N.^2+M.*N.^2)+224.*g2.*g4.^2.*N.^2.*((-4)+(-2).*M+2.*N+ ...
M.*N+N.^2+M.*N.^2)+64.*g2.*g3.*g4.*N.^2.*((-14)+(-4).*M+7.*N+2.* ...
M.*N+6.*N.^2+M.*N.^2)+64.*g1.*g3.*g4.*N.^2.*((-14)+(-4).*M+7.*N+ ...
2.*M.*N+2.*N.^2+2.*M.*N.^2)+8.*g2.*g3.^2.*N.^2.*((-32)+(-4).*M+ ...
16.*N+2.*M.*N+9.*N.^2+2.*M.*N.^2)+16.*g1.*g2.*g3.*N.*(80+16.*M+( ...
-12).*N+7.*N.^2+2.*M.*N.^2+N.^3)+8.*g1.^2.*g3.*N.*(80+16.*M+(-12) ...
.*N+4.*N.^2+2.*M.*N.^2+3.*N.^3+N.^4)+4.*g2.^2.*g3.*N.*(160+32.*M+( ...
-24).*N+17.*N.^2+7.*M.*N.^2+4.*N.^3+N.^4)+32.*g1.*g2.*g4.*N.*(96+ ...
36.*M+(-24).*N+(-12).*M.*N+2.*N.^2+(-2).*M.*N.^2+4.*N.^3+M.*N.^3+ ...
N.^4)+4.*g1.^3.*((-160)+(-48).*M+40.*N+12.*M.*N+4.*N.^2+2.*M.* ...
N.^2+4.*N.^3+M.*N.^3+2.*N.^4)+2.*g1.*g2.^2.*((-960)+(-288).*M+ ...
240.*N+72.*M.*N+(-88).*N.^2+(-20).*M.*N.^2+39.*N.^3+7.*M.*N.^3+ ...
13.*N.^4)+16.*g1.^2.*g4.*N.*(96+36.*M+(-24).*N+(-12).*M.*N+(-4).* ...
N.^2+(-2).*M.*N.^2+2.*N.^3+3.*M.*N.^3+M.*N.^4)+8.*g1.^2.*g2.*(( ...
-240)+(-72).*M+60.*N+18.*M.*N+(-8).*N.^2+(-1).*M.*N.^2+6.*N.^3+2.* ...
M.*N.^3+N.^4+M.*N.^4)+8.*g2.^2.*g4.*N.*(192+72.*M+(-48).*N+(-24).* ...
M.*N+10.*N.^2+2.*M.*N.^2+6.*N.^3+4.*M.*N.^3+N.^4+M.*N.^4)+g2.^3.*( ...
(-640)+(-192).*M+160.*N+48.*M.*N+(-96).*N.^2+(-24).*M.*N.^2+36.* ...
N.^3+12.*M.*N.^3+10.*N.^4+5.*M.*N.^4)).*pi.^(-2);
```

18.6 Define the system

```
dg1_dt = -epsilon*g1 + beta12(M,N);
dg2_dt = -epsilon*g2 + beta22(M,N);
dg3_dt = -epsilon*g3 + beta32(M,N);
dg4_dt = -epsilon*g4 + beta42(M,N);
system = [dg1_dt; dg2_dt; dg3_dt; dg4_dt];
```

In general there are no modifications needed after this line.

18.7 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...   % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...      % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

18.8 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

NINETEEN

HOMOCLINIC RG FLOWS

In [JP21] an $\mathcal{N} = 1$ supersymmetric model of interacting scalar superfields that is invariant under the action of an $O(N) \times O(M)$ group in $d = 3 - \epsilon$ dimensions is considered. The coupling constants $g_i (i = 1, \dots, 4)$ satisfy the following differential equations

$$\dot{g}_i = -\epsilon g_i + \beta_i^{(2)}, \qquad i = 1, \dots, 4,$$

where the β functions are given by

$$\begin{aligned}\beta_1^{(2)} =& \frac{1}{8\pi^2 N^2} \left(32 g_1^2 g_4 N \left(-40 - 8M + 8N + 7MN + 8N^2\right) + 16 g_1^2 g_3 N \left(-80 - 24M + 30N + 4MN + 7N^2 + 4MN^2\right)\right. \\ & + 16 g_1 g_3^2 N^2 \left(32 + 2M + N + MN + N^2 + MN^2\right) + 64 g_1 g_2 g_4 N \left(-32 - 4M + 16N + 2MN + 5N^2 + 2MN^2\right) \\ & + 4 g_2^2 g_3 N \left(-256 - 64M + 72N + 10MN + 19N^2 + 2MN^2\right) + 64 g_1 g_4^2 N^2 \left(22 - 2M + MN + MN^2\right) \\ & + 16 g_1 g_2 g_3 N \left(-144 - 40M + 42N + 17MN + 21N^2 + 5MN^2\right) + 128 g_1 g_3 g_4 N^2 \left(3 + 5M + N + N^2\right) \\ & + g_2^3 \left(896 + 128M - 352N - 48MN - 12N^2 - 12MN^2 + 7N^3 + 2MN^3\right) + 96 g_2^2 g_4 (-8 + N) N \\ & + 4 g_1^2 g_2 \left(928 + 224M - 392N - 100MN - 2N^2 + 11MN^2 + 23N^3 + 7MN^3 + 4N^4\right) + 768 g_2 g_3 g_4 N^2 \\ & + 4 g_1^3 \left(352 + 96M - 152N - 44MN + 10N^3 + 3MN^3 + MN^4\right) + 16 g_2 g_3^2 N^2 \left(16 + 7M + N + N^2\right) \\ & \left. + 2 g_1 g_2^2 \left(1600 + 320M - 656N - 136MN + 32N^2 + 10MN^2 + 50N^3 + 10MN^3 + 5N^4 + 2MN^4\right)\right),\end{aligned}$$

$$\begin{aligned}\beta_2^{(2)} =& \frac{1}{8\pi^2 N^2} \left(64 g_1 g_2 g_4 N \left(-80 - 16M + 16N + 5MN + 7N^2\right) + 16 g_2 g_3^2 N^2 \left(48 + 9M + 2N + MN + 2N^2 + MN^2\right)\right. \\ & + 16 g_1^2 g_3 N \left(-128 - 32M + 32N + 12MN + 14N^2 + 3MN^2\right) + 128 g_2 g_3 g_4 N^2 \left(9 + 5M + N + N^2\right) \\ & + 16 g_1 g_2 g_3 N \left(-272 - 72M + 82N + 15MN + 24N^2 + 6MN^2\right) + 192 g_1^2 g_4 N \left(-12 - 2M + 4N + N^2\right) \\ & + 16 g_2^2 g_4 N \left(-176 - 40M + 58N + 14MN + 17N^2 + 11MN^2\right) + 32 g_1 g_3^2 N^2 \left(16 + 7M + N + N^2\right) \\ & + 4 g_2^2 g_3 N \left(-576 - 160M + 176N + 54MN + 74N^2 + 17MN^2\right) + 64 g_2 g_4^2 N^2 \left(22 - 2M + MN + MN^2\right) \\ & + 8 g_1^3 \left(288 + 64M - 112N - 24MN - 6N^2 + 3MN^2 + 5N^3 + 2MN^3 + N^4\right) + 1536 g_1 g_3 g_4 N^2 \\ & + 2 g_1 g_2^2 \left(3968 + 1024M - 1600N - 416MN - 56N^2 - 22MN^2 + 85N^3 + 17MN^3 + 11N^4\right) \\ & + 4 g_1^2 g_2 \left(1856 + 448M - 736N - 176MN - 22N^2 - 5MN^2 + 43N^3 + 8MN^3 + 3N^4 + 2MN^4\right) \\ & \left. + g_2^3 \left(2816 + 768M - 1152N - 320MN + 12N^2 - 12MN^2 + 69N^3 + 30MN^3 + 7N^4 + 5MN^4\right)\right),\end{aligned}$$

$$
\begin{aligned}
\beta_3^{(2)} =& \frac{1}{8\pi^2 N^3}\left(32 g_3^2 g_4 N^3\left(18+14M+7N+7N^2\right)+96 g_1^2 g_4 N\left(8+2N^2+MN^2\right)+384 g_1 g_2 g_4 N\left(4+N^2\right)\right.\\
&+24 g_3^3 N^3\left(16+2M+2N+MN+2N^2+MN^2\right)+64 g_2 g_3 g_4 N^2\left(-20-4M+10N+2MN+5N^2+2MN^2\right)\\
&+16 g_1 g_3^2 N^2\left(-52-14M+26N+7MN+14N^2+7MN^2\right)+128 g_1 g_3 g_4 N^2\left(-10-2M+5N+MN+5N^2\right)\\
&+8 g_2 g_3^2 N^2\left(-104-28M+52N+14MN+38N^2+7MN^2\right)+64 g_3 g_4^2 N^3\left(22-2M+MN+MN^2\right)\\
&+16 g_1 g_2 g_3 N\left(208+56M-48N-12MN+12N^2+3MN^2+8N^3+MN^3+2N^4\right)+96 g_2^2 g4 N\left(8+N^2\right)\\
&+8 g_1^2 g_3 N\left(208+56M-48N-12MN+12N^2+6MN^2+4N^3+3MN^3+MN^4\right)\\
&+8 g_1^3\left(-96-16M+24N+4MN-20N^2-10MN^2+6N^3+2MN^3+N^4+MN^4\right)\\
&+4 g_2^2 g_3 N\left(416+112M-96N-24MN+18N^2+6MN^2+12N^3+4MN^3+2N^4+MN^4\right)\\
&+g_2^3\left(-768-128M+192N+32MN-96N^2+32N^3+4MN^3+7N^4+2MN^4\right)\\
&+2 g_1 g_2^2\left(-1152-192M+288N+48MN-176N^2-40MN^2+70N^3+10MN^3+21N^4+2MN^4\right)\\
&\left.+4 g_1^2 g_2\left(-576-96M+144N+24MN-104N^2-40MN^2+34N^3+13MN^3+11N^4+3MN^4\right)\right),
\end{aligned}
$$

and

$$
\begin{aligned}
\beta_4^{(2)} =& \frac{1}{8\pi^2 N^3}\left(8 g_1^2 g_3 N\left(80+16M-12N+4N^2+2MN^2+3N^3+N^4\right)\right.\\
&+224 g_1 g_4^2 N^2\left(-4-2M+2N+MN+2N^2\right)+16 g_1 g_3^2 N^2\left(-16-2M+8N+MN+7N^2\right)\\
&+96 g_4^3 N^3\left(8-2M+MN+MN^2\right)+64 g_2 g_3 g_4 N^2\left(-14-4M+7N+2MN+6N^2+MN^2\right)\\
&+224 g_2 g_4^2 N^2\left(-4-2M+2N+MN+N^2+MN^2\right)+32 g_3^2 g_4 N^3\left(22+N+MN+N^2+MN^2\right)\\
&+64 g_1 g_3 g_4 N^2\left(-14-4M+7N+2MN+2N^2+2MN^2\right)\\
&+8 g_2 g_3^2 N^2\left(-32-4M+16N+2MN+9N^2+2MN^2\right)+24 g_3^3 N^3\left(4+2M+N+N^2\right)\\
&+16 g_1 g_2 g_3 N\left(80+16M-12N+7N^2+2MN^2+N^3\right)+224 g_3 g_4^2 N^3\left(2M+N+N^2\right)\\
&+4 g_2^2 g_3 N\left(160+32M-24N+17N^2+7MN^2+4N^3+N^4\right)\\
&+32 g_1 g_2 g_4 N\left(96+36M-24N-12MN+2N^2-2MN^2+4N^3+MN^3+N^4\right)\\
&+4 g_1^3\left(-160-48M+40N+12MN+4N^2+2MN^2+4N^3+MN^3+2N^4\right)\\
&+2 g_1 g_2^2\left(-960-288M+240N+72MN-88N^2-20MN^2+39N^3+7MN^3+13N^4\right)\\
&+16 g_1^2 g_4 N\left(96+36M-24N-12MN-4N^2-2MN^2+2N^3+3MN^3+MN^4\right)\\
&+8 g_1^2 g_2\left(-240-72M+60N+18MN-8N^2-MN^2+6N^3+2MN^3+N^4+MN^4\right)\\
&+8 g_2^2 g_4 N\left(192+72M-48N-24MN+10N^2+2MN^2+6N^3+4MN^3+N^4+MN^4\right)\\
&\left.+g_2^3\left(-640-192M+160N+48MN-96N^2-24MN^2+36N^3+12MN^3+10N^4+5MN^4\right)\right).
\end{aligned}
$$

The parameter ϵ is fixed to 1, while M and N are taken as unfolding parameters.

19.1 Overview

In this demo we will use the new homoclinic predictor from [Kuz21] to continue homoclinic curves from generic Bogdanov-Takens points. In order to do this we will:

- Compute a curve of equilibria, parametrized by M.
- Detect various limit and Hopf points.
- Start continuation from one of the detected Hopf points in two parameters (M, N).
- Detect two Bogdanov-Takens points.
- Start continuation from the Bogdanov-Takens points in two parameters (M, N).
- Compare the predicted and computed homoclinic bifurcation curve emanating from the first the Bogdanov-Takens point in parameters space.

- Compare a range of predictors for the homoclinic solutions emanating from the first Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space.
- Create bifurcation plots including Hopf and fold curves.
- Create a convergence plot comparing the different homoclinic approximations derived in [Kuz21].

19.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Equilibrium'])
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
```

19.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *HomoclinicRGflowsGenSym.ipynb*.

```
odefile=@HomoclinicRGflows;
```

19.4 Define equilibrium

We manually define an equilibrium at

$$(g_1, g_2, g_3, g_4) = (0.27495712275636564, 1.3931601076374327, -0.30951743797410936, -0.30951743797410936), \tag{1}$$

with parameter values $M = 0.2945$ and $N = 4.036$.

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in the software package *DDE-BifTool* [SEL+14].

```
parnames = {'epsilon', 'M', 'N'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
```

(continues on next page)

(continued from previous page)

```
p(ind.epsilon) = 1;
p(ind.M) = 0.2945;
p(ind.N) = 4.036;
x  = [0.0701457361241472, -0.06520883770451065, 0.001823543197553845, 0.
↪22874527306411319]';
```

19.5 Continue equilibrium in parameter N

To continue the equilibrium (1) in parameter N, we first need to obtain a tangent vector to the curve. This is done by the function `init_EP_EP`. Then we use the function `contset` to obtain a **struct** containing a list of options which is passed on to the continuer. By adjusting the values of the fields of the `opt` **struct** we set the maximum step size. We also set the maximum number of points to continue and weather or not to detect bifurcation points (`opt.Singularities`) on the equilibrium curve. For more information about all options available to the *MatCont* continuer and the continuation process in general, we refer to [DGK+08].

Finally, we continue the curve using the function `cont`.

```
[x1_pred, v1_pred] = init_EP_EP(odefile, x, p, ind.M);
opt = contset;
opt.MaxNumPoints  = 300;
opt.Singularities = 1;
opt.Backward = 1;
[eqbr_x, ~, eqbr_bif_data] = cont(@equilibrium, x1_pred, v1_pred, opt);
```

```
first point found
tangent vector to first point found
Neutral saddle
label = H , x = ( 0.000002 -0.000003 -0.000000 0.171390 1.091355 )
label = LP, x = ( -0.110189 0.142208 0.003572 0.100776 1.998164 )
a=-7.331116e-01
Neutral saddle
label = H , x = ( -0.579707 0.570669 0.013951 -0.019833 0.966823 )
label = LP, x = ( -0.876594 0.748784 0.043787 -0.092572 0.823874 )
a=3.488423e+00
Neutral saddle
label = H , x = ( -0.945788 0.778645 0.058703 -0.115668 0.837025 )
Neutral saddle
label = H , x = ( -0.990782 0.796522 0.076078 -0.141243 0.873995 )
label = LP, x = ( -1.273488 1.256015 0.469461 -0.638409 1.000164 )
a=7.585356e-01
label = LP, x = ( -1.345207 1.354790 0.724609 -0.959767 0.979759 )
a=-8.763553e-01
Neutral saddle
label = H , x = ( -1.042702 1.041690 0.603050 -0.814403 0.981600 )
label = BP, x = ( -0.541043 0.530513 0.349786 -0.520872 1.001623 )
label = BP, x = ( -0.541043 0.530513 0.349786 -0.520872 1.001623 )
Neutral saddle
label = H , x = ( -0.118398 0.237762 0.080743 -0.300349 1.024672 )
label = LP, x = ( -0.009751 0.027361 0.006892 -0.184779 1.098410 )
a=4.606024e-02
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
↪RCOND =  1.832631e-16.
> In equilibrium>locateBP (line 300)
```

(continues on next page)

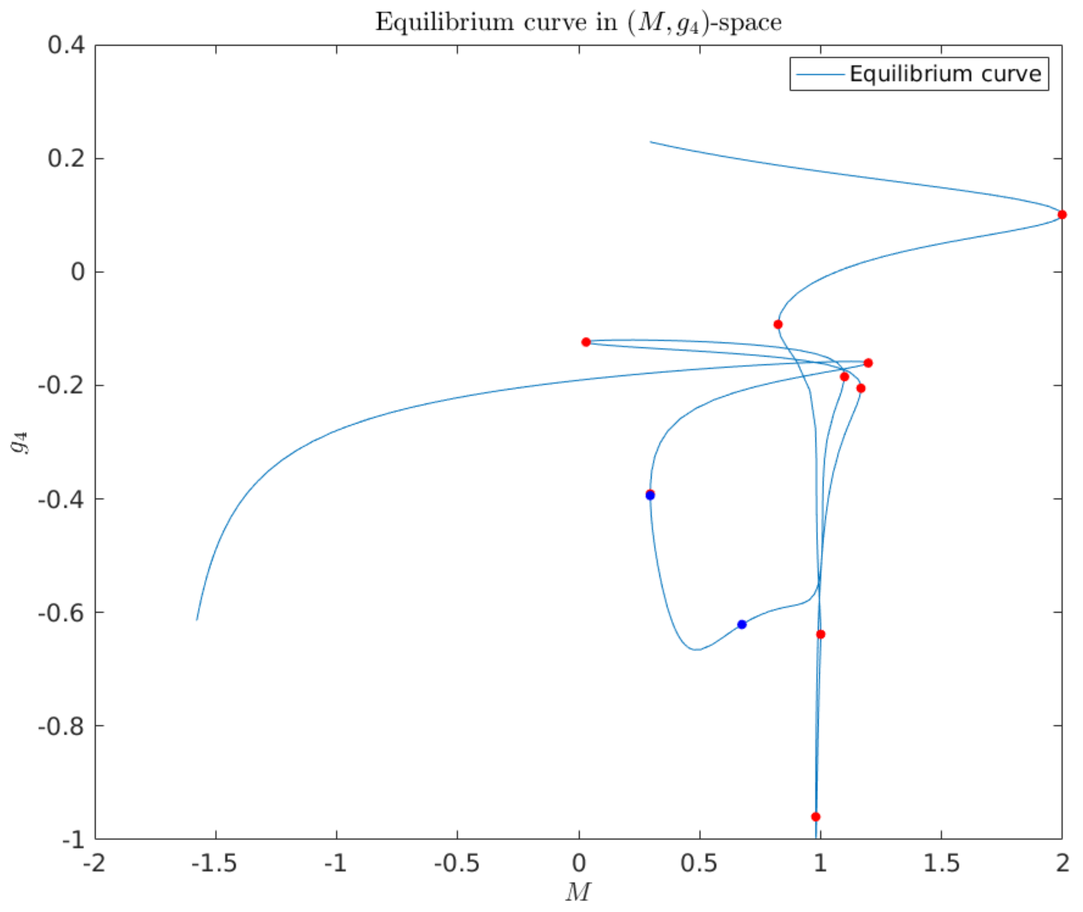
(continued from previous page)

```
In equilibrium>locate (line 214)
In cont (line 454)

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
↪RCOND =  1.671679e-16.
> In equilibrium>locateBP (line 300)
In equilibrium>locate (line 214)
In cont (line 454)

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
↪RCOND =  1.796148e-16.
> In equilibrium>locateBP (line 300)
In equilibrium>locate (line 214)
In cont (line 454)

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
↪RCOND =  6.596977e-17.
> In equilibrium>locateBP (line 300)
In equilibrium>locate (line 214)
In cont (line 454)

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
↪RCOND =  9.044303e-18.
> In equilibrium>locateBP (line 300)
In equilibrium>locate (line 214)
In cont (line 454)

Neutral saddle
label = H , x = ( 0.000004 -0.000012 -0.000003 -0.171382 1.091381 )
label = LP, x = ( 0.082791 -0.165490 -0.069470 -0.123735 0.030515 )
a=1.327431e+00
Neutral saddle
label = H , x = ( 0.017075 -0.016965 -0.013449 -0.162484 1.015832 )
Neutral saddle
label = H , x = ( 0.000034 -0.000033 -0.000027 -0.171369 1.091270 )
label = LP, x = ( -0.057200 0.053194 0.043292 -0.205042 1.166392 )
a=1.588963e-01
label = H , x = ( -1.161474 -0.089034 0.877004 -0.621034 0.674701 )
First Lyapunov coefficient = 4.229695e-01
label = H , x = ( -0.720879 -0.252515 0.514609 -0.394150 0.295843 )
First Lyapunov coefficient = -3.941504e+00
label = LP, x = ( -0.712373 -0.249398 0.507987 -0.391094 0.295815 )
a=-6.258319e+02
Neutral saddle
label = H , x = ( -0.386826 -0.127182 0.261001 -0.280979 0.367126 )
label = LP, x = ( 0.065249 0.011257 -0.036715 -0.160519 1.196516 )
a=-2.872173e-01
Neutral saddle
label = H , x = ( 0.648955 0.299168 -0.444774 -0.280762 -1.007194 )

elapsed time  = 1.2 secs
npoints curve = 300
```

There are multiple Hopf (H) and limit bifurcation points detected (LP). The **array struct** `eqbr_bif_data` contains information about the detected bifurcation points. We use this to extract the index of the detected bifurcation points on the equilibrium curve `eqbr_x`. The equilibrium curve `eqbr_x` is just a two dimensional array. Each column consists of a point on the curve. The first four rows contain the point g while the last row contains the parameter M.

Below we plot the equilibrium curve `eqbr_x`, together with the detected Hopf and limit points, in (M, g_4)-space.

```
%plot --width 1024 --height 800
plot(eqbr_x(5,:), eqbr_x(4,:)); hold on
foldInfo    = eqbr_bif_data(strcmp({eqbr_bif_data.msg}, 'Limit point')==1);
foldInfocell = struct2cell(foldInfo);
foldInd = cell2mat(foldInfocell(1,:));
hopfInfo    = eqbr_bif_data(strcmp({eqbr_bif_data.msg}, 'Hopf')==1);
hopfInfocell = struct2cell(hopfInfo);
hopfInd = cell2mat(hopfInfocell(1,:));
plot(eqbr_x(5,foldInd), eqbr_x(4,foldInd), '.r', 'MarkerSize', 20); hold on
plot(eqbr_x(5,hopfInd), eqbr_x(4,hopfInd), '.b', 'MarkerSize', 20); hold on
xlabel('$M$')
ylabel('$g_4$')
legend({'Equilibrium curve'}, 'Location', 'NorthEast')
title('Equilibrium curve in $(M,g_4)$-space')
```



19.6 Setup Hopf point

To continue the first Hopf point detected on the equilibrium branch `eqbr_x` in the parameters M and N we construct a new point `Hopf` containing the position and parameter values. These are needed to obtain an initial tangent vector - using the function `init_H_H` - in the full phase/parameter space. Since, from now on, we will be using the continuation parameters M and N frequently we assigned these parameters to the variable `ap` (active parameters).

```
ap = [ind.M ind.N];
hopfInfo = eqbr_bif_data(strcmp({eqbr_bif_data.msg}, 'Hopf')==1);
hopf.x = eqbr_x(1:4,hopfInfo(2).index);
hopf.par = p';
hopf.par(ind.M) = eqbr_x(5,hopfInfo(2).index);
[hopf1_x, hopf1_v] = init_H_H(odefile, hopf.x, hopf.par, ap);
```

19.7 Continue Hopf point in parameters M and N

We continue the Hopf point curve using again the function `cont`. We use the same continuation options as before defined above in the **struct** `opt`, but set additionally the following options. We increase the number of maximum allowed continuation points. We also increase the accuracy for locating detected bifurcations (`TestTolerance`) and the maximum number of iterations that may be used to achieve this (`MaxTestIters`). This improves the homoclinic predictor which depend directly on the accuracy of the located Bogdanov-Takens point.

```
opt.TestTolerance = 1e-12;
opt.MaxTestIters = 10;
opt.Backward = 0;
opt.MaxNumPoints = 50;
[hopf_br, ~, hopf_br_bif] = cont(@hopf, hopf1_x, hopf1_v, opt);
```

```
first point found
tangent vector to first point found
label = BT, x = ( -0.715157 -0.250968 0.510051 -0.391935 0.294477 4.035536 0.000000 )
(a,b)=(1.387256e+00, -1.894680e-01)
label = HH, x = ( -0.642016 -0.228678 0.452440 -0.364578 0.283522 4.030314 -0.291003 )
Neutral saddle ?
label = BT, x = ( -0.133317 0.022458 0.090992 -0.220100 0.883754 4.419949 0.000000 )
(a,b)=(-2.411266e-02, -5.207995e-01)
label = BT, x = ( -0.132423 0.106649 0.089399 -0.248054 0.942195 4.593759 -0.000000 )
(a,b)=(-7.110064e-03, 5.333636e-02)
label = HH, x = ( -0.410202 0.398065 0.256144 -0.441341 0.950248 4.384951 -0.038318 )
Neutral saddle ?

elapsed time  = 0.5 secs
npoints curve = 50
```

There are two Bogdanov-Takens bifurcation points (BT) detected on the limit point branch `lp_br`.

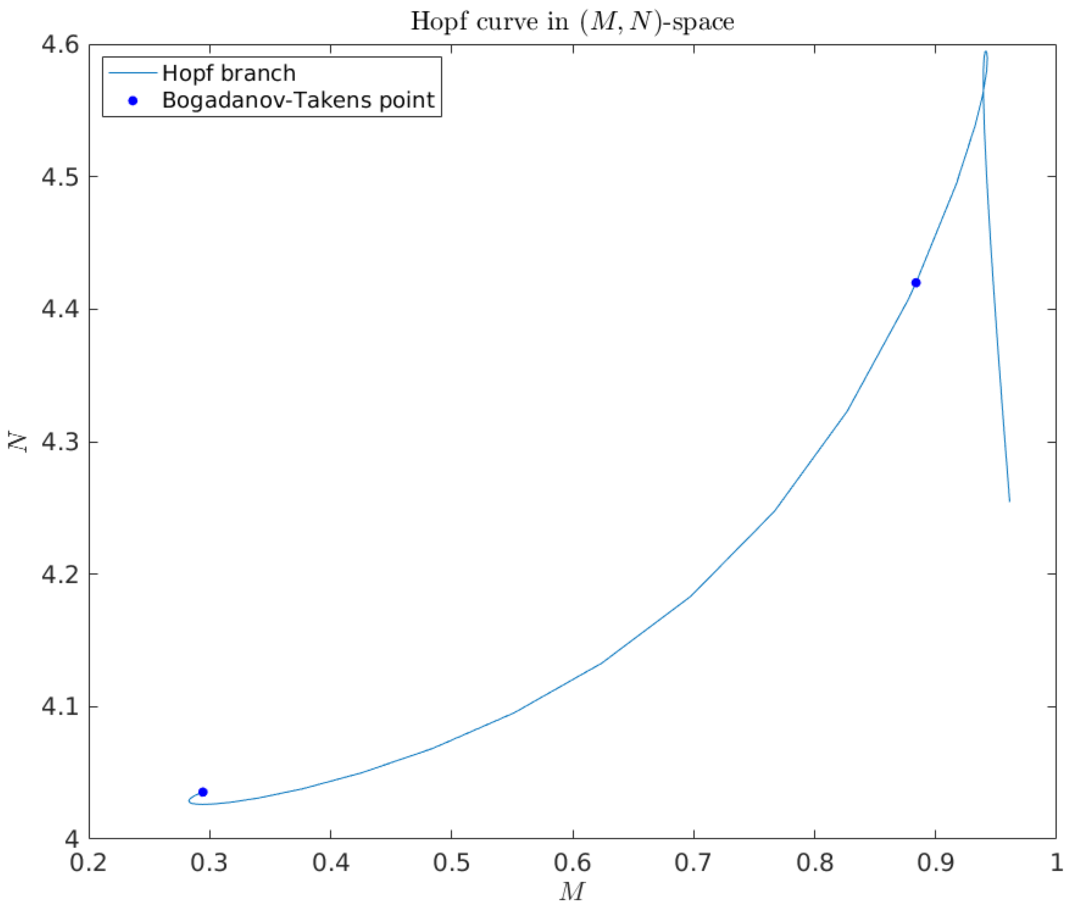
As with the limit points, information about the detected bifurcation points is stored in the **struct array** `lp_br_bif`. Below we extract the Bogdanov-Takens bifurcation points.

```
bt_points_info = hopf_br_bif(strcmp({hopf_br_bif.label}, 'BT')==1);
BTPoint1 = hopf_br(:,bt_points_info(1).index);
BTPoint2 = hopf_br(:,bt_points_info(2).index);
plot(hopf_br(5,:), hopf_br(6,:)); hold on
plot(BTPoint1(5), BTPoint1(6), '.b' ,'MarkerSize', 20)
```

(continues on next page)

(continued from previous page)

```
plot(BTPoint2(5), BTPoint2(6), '.b' ,'MarkerSize', 20)
xlabel('$M$')
ylabel('$N$')
legend({'Hopf branch', 'Bogadanov-Takens point'}, 'Location', 'NorthWest')
title('Hopf curve in $(M,N)$-space')
```



19.8 Initial prediction of homoclinic orbit near Bogdanov-Takens point 1

To obtain an initial approximation to the homoclinic solution near the Bogdanov-Takens point we use the function `init_BT_Hom`. Its arguments are the system file (`odefile`), the Bogdanov-Takens point (`bt1`) as defined below, the unfolding parameters (`ap`) and an options structure (`BToptions`). The options structure created with the function `BT_Hom_set_options` contains the following fields:

- `ntst` Number of mesh intervals with a default value of 40.
- `ncol` Number of collocation points used in each interval with a default of 4.
- `extravec` Three dimensional boolean row vector indicating which *homoclinic parameters* are selected to be free. The first component refers to the half-return time, while the second and third components refer to the distances

from the saddle point to the first, respectively, the last point on the homoclinic orbit. The default value is set to `[0 1 1]`. Thus, the half-return time `T` is fixed.

- `order` The order of the homoclinic approximation used with a default value of 3.
- `amplitude` Desired amplitude of the homoclinic solution. If left empty then a conservative estimate is made, see [Kuz21].
- `TTolerance` Desired distance between the last point on the numerical homoclinic solution and the saddle point. This should be at least be smaller than the amplitude. If left empty it is defined by `amplitude*1.0e-03`.
- `HigherOrderTimeReparametrization` Boolean to indicate if a higher order approximation to the nonlinear time transformation in the Lindstedt-Poincaré method should be used. This should always be set to `1`. It is only implemented for demonstration purposes.
- `method` Selects the method to be used to approximate the homoclinic solution. The different methods available are:
  - orbital (the default),
  - orbitalv2,
  - LP (Lindstedt-Poincaré with smooth normal form),
  - LPHypernormalForm,
  - RegularPerturbation,
  - RegularPerturbationL2.

  We refer to [Kuz21] for the interpretations.
- `messages` Boolean to indicate if information about selected parameter should be printed the console. The default value is set to `true`.
- `correct` Boolean to indicate if the predicted homoclinic solution should be corrected with Newton. The default value is set to `true`.

Here we will use most of of default values for the Bogdanov-Takens option structure. We set the field `correct` to `false` and manually correct the approximation. Also, we set the field `amplitude` to `0.2` to start continuation closer to the Bogdanov-Takens point. This looks better in the bifurcation diagram below. However, if we do not set the field `amplitiude` convergence is achived aswell.

```
bt_index = bt_points_info(1).index;
bt1.x = hopf_br(1:4, bt_index);
bt1.par = p';
bt1.par(ap) = hopf_br(5:6, bt_index);
BToptions = BT_Hom_set_options();
BToptions.correct = false;
BToptions.amplitude = 0.2;
[x1_pred, v1_pred] = init_BT_Hom(odefile, bt1, ap, BToptions);
```

```
Center manifold coefficients' accuracy: 5.456968e-12
BT normal form coefficients:
a=1.387256e+00,         b=-1.894680e-01
The initial perturbation parameter epsilon:  2.936952e-02
The initial amplitude: 0.2
The initial half-return time T: 19.6487
The initial distance eps0: 0.000170201
The initial distance eps1: 0.000126381
```

19.9 Correct initial prediction of homoclinic orbit near bt1 with Newton

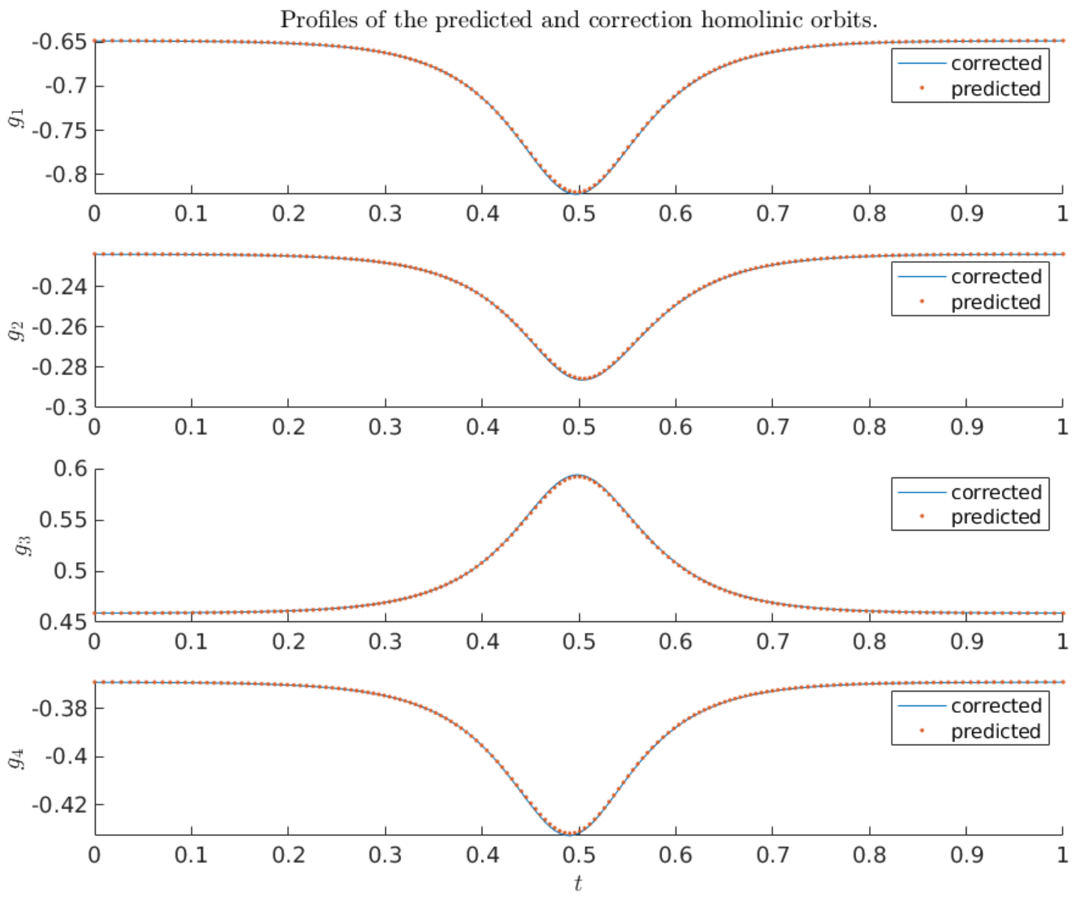
Now that we have an initial prediction for the homoclinic orbit we manually correct it using Newton. After the homoclinic predictor is corrected with the **MatCont** function `newtcorr` we use the function `bt_rearr` (Bogdanov-Takens rearrange) to extract the homoclinic orbit and saddle point from the homoclinic correction.

```
[hom1_x, hom1_v, ~] = newtcorr(x1_pred, v1_pred);
[x1_orbit, x1_saddle] = bt_rearr(hom1_x);
```

19.10 Compare profiles of predicted and corrected solution (bt1)

Using again the **MatCont** function `bt_rearr`, but now on the homoclinic prediction `x1_pred` we compare the profiles of the predicted and corrected homoclinic orbits. We see that they are indistinguishable. Note that to access the mesh on which the homoclinic orbit is computed we need the global variable `homds`.

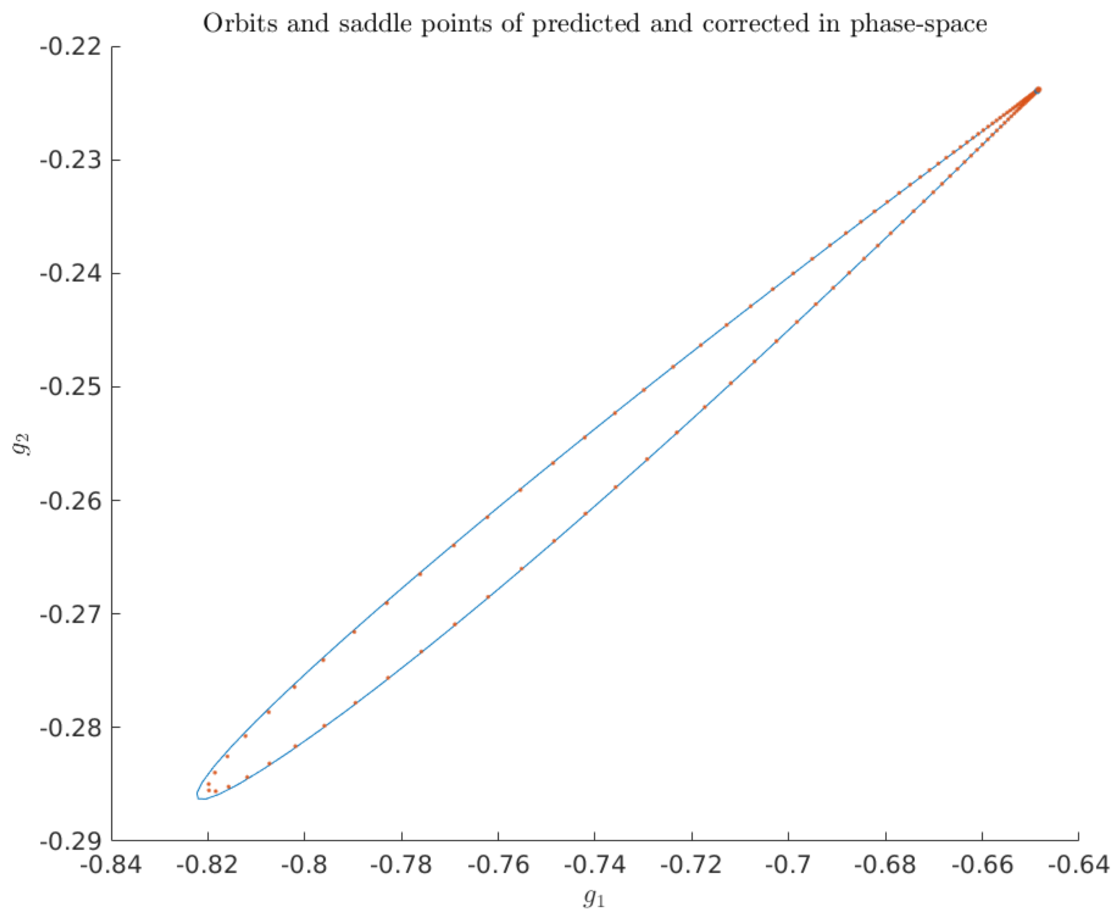
```
[homoclinic1_pred, saddle1_pred] = bt_rearr(x1_pred);
subplot(4,1,1); hold on;
global homds
title('Profiles of the predicted and correction homolinic orbits.')
plot(homds.finemsh, x1_orbit(1:4:end))
plot(homds.finemsh, homoclinic1_pred(1:4:end),'.')
legend({'corrected', 'predicted'})
ylabel('$g_1$')
subplot(4,1,2); hold on;
plot(homds.finemsh, x1_orbit(2:4:end))
plot(homds.finemsh, homoclinic1_pred(2:4:end),'.')
legend({'corrected','predicted'})
ylabel('$g_2$')
subplot(4,1,3); hold on;
plot(homds.finemsh, x1_orbit(3:4:end))
plot(homds.finemsh, homoclinic1_pred(3:4:end),'.')
legend({'corrected','predicted'})
ylabel('$g_3$')
subplot(4,1,4); hold on;
plot(homds.finemsh, x1_orbit(4:4:end))
plot(homds.finemsh, homoclinic1_pred(4:4:end),'.')
legend({'corrected','predicted'})
ylabel('$g_4$')
legend({'corrected',  'predicted'})
xlabel('$t$')
```



19.11 Compare predictor and corrected solution in (g_1, g_2) phase-space

Below we compare the predicted and corrected homoclinic orbit in (g_1, g_2) phase-space, as well as the predicted and corrected saddle point.

```
hold on
plot(x1_orbit(1:4:end),x1_orbit(2:4:end))
plot(homoclinic1_pred(1:4:end),homoclinic1_pred(2:4:end),'.')
plot(x1_saddle(1), x1_saddle(2),'.', 'MarkerSize', 12, 'Color', [0 0.4470 0.7410])
plot(saddle1_pred(1), saddle1_pred(2),'.', 'MarkerSize', 12, 'Color', [0.8500, 0.3250,
 ↪ 0.0980])
xlabel('$g_1$')
ylabel('$g_2$')
title('Orbits and saddle points of predicted and corrected in phase-space')
```



19.12 Continue homoclinic curve emanating from the first Bogdanov-Takens point

Having obtain an initial approximation `[hom_x, hom_v]`, where `homo_v` is the tangent vector to the homoclinic curve pointing outwards from the Bogdanov-Takens point, we can start continuation using the function `cont`.

```
[homoclinic_br1, homoclinic_br1_v, homoclinic_singularities] = cont(@homoclinic, hom1_
↪x, hom1_v, opt);
```

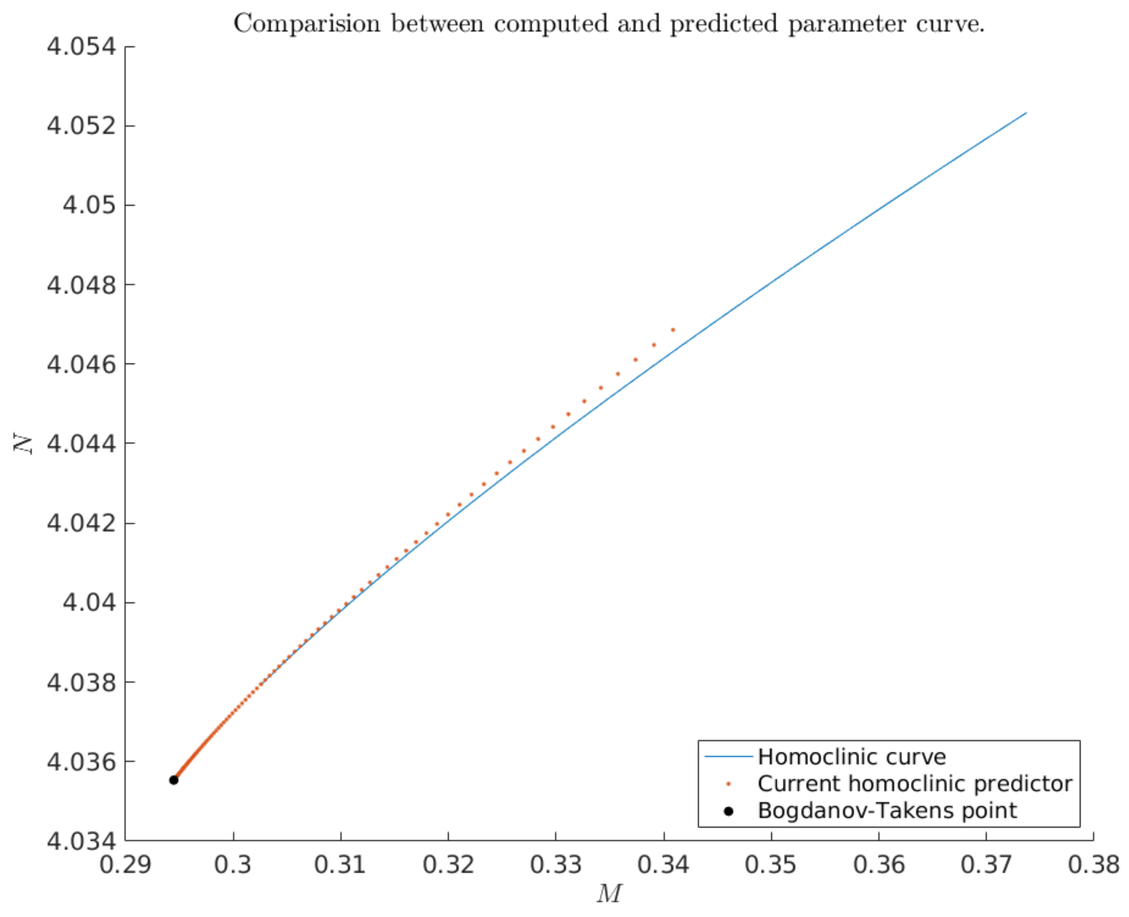
```
first point found
tangent vector to first point found
Inclination-flip with respect to the unstable manifold, parameters = 0.32798 and 4.
↪04374.

elapsed time  = 8.1 secs
npoints curve = 50
```

19.13 Compare predicted with computed parameters emanating from bt1

Now that we have obtained a curve of homoclinic orbits (`homoclinic_br`) we compare the computed curve in parameter space with the predicted curve we construct below. To do so, we use the function `BT_nmfm_orbital` to obtain the smooth orbital normal form coefficients, i.e. a and b, and the coefficients of the transformation K between the parameters of the system and the parameters in the smooth orbital normal form, see [Kuz21].

```
hold on
% plot computed homoclinic parameter curve
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:));
% Bogdanov-Takens parameter-dependent smooth orbital normal form coefficients
bt1 = BT_nmfm_orbital(odefile, bt1, ap, BToptions);
a   = bt1.nmfm.a;
b   = bt1.nmfm.b;
K10 = bt1.nmfm.K10;
K01 = bt1.nmfm.K01;
K02 = bt1.nmfm.K02;
K11 = bt1.nmfm.K11;
K03 = bt1.nmfm.K03;
% construct predictor as in the paper
eps = linspace(0, 0.05);
beta1 = -4*a^3/b^4*eps.^4;
tau0  = 10/7;
tau2  = 288/2401;
beta2 = a/b*(tau0 + tau2*eps.^2).*eps.^2;
alpha = K10.*beta1 + K01.*beta2 + 1/2*K02.*beta2.^2 ...
    + K11.*beta1.*beta2 + 1/6*K03.*beta2.^3;
alpha = bt1.par(ap) + alpha;
% plot currect predictor
plot(alpha(1,:), alpha(2,:), '.')
% plot Bogdanov-Takens point
plot(bt1.par(ind.M), bt1.par(ind.N), '.k', 'MarkerSize', 20)
% set axis labels and legend
xlabel('$M$')
ylabel('$N$')
legend({'Homoclinic curve', 'Current homoclinic predictor', ...
    'Bogdanov-Takens point'}, 'Location', 'SouthEast')
title('Comparision between computed and predicted parameter curve.')
```

```
Center manifold coefficients' accuracy: 5.456968e-12
```



19.14 Bifurcation diagram in (g_1, g_2, g_3) phase-space

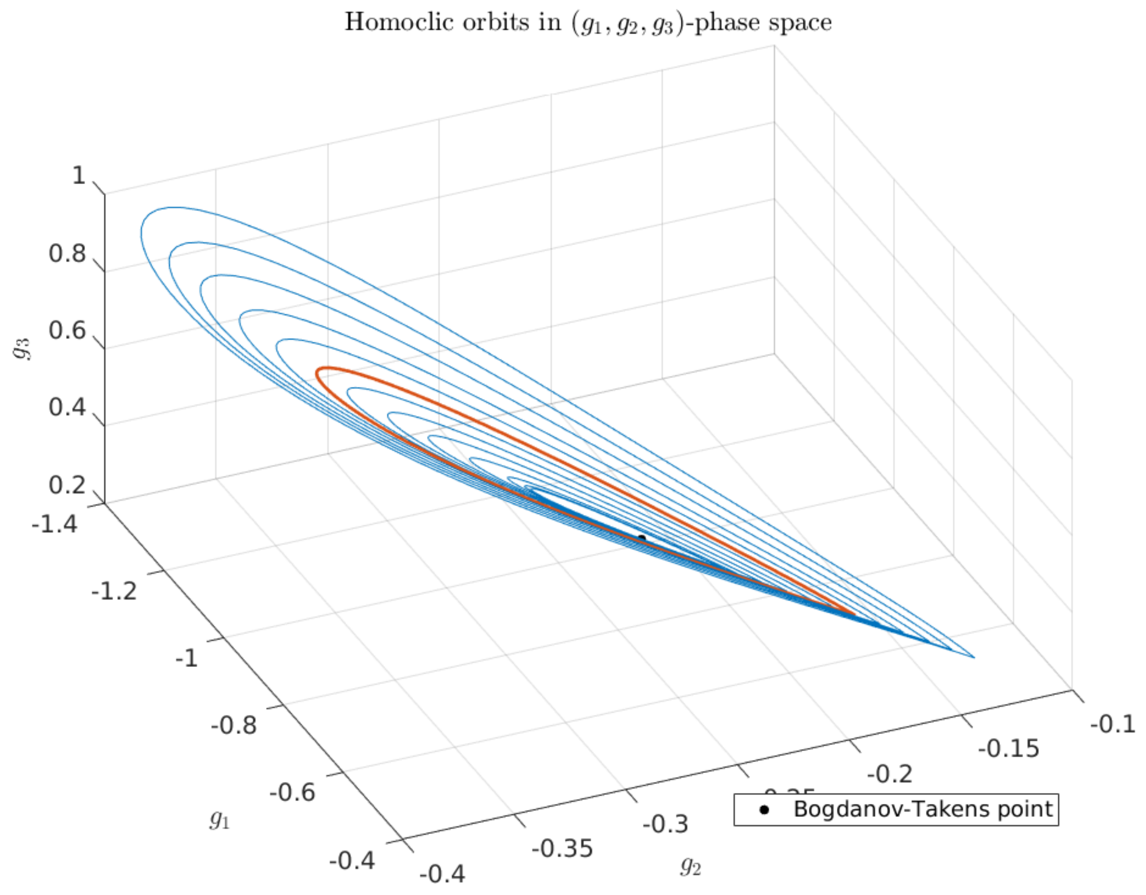
To obtain an impression of the homoclinic solutions we plot the computed homoclinic orbits in (g_1, g_2, g_3) phase-space. The red curve is the singularities detected on the homoclinic branch.

```
global homds
cm = lines;
hold on
plot3(homoclinic_br1(homds.coords(1:homds.nphase:end), 1:4:end), ...
      homoclinic_br1(homds.coords(2:homds.nphase:end), 1:4:end), ...
      homoclinic_br1(homds.coords(3:homds.nphase:end), 1:4:end), ...
      'Color', cm(1,:), 'HandleVisibility', 'Off')
bif_points = struct2cell(homoclinic_singularities);
plot3(homoclinic_br1(homds.coords(1:homds.nphase:end), cell2mat(bif_points(1,2:end-
↪1))), ...
      homoclinic_br1(homds.coords(2:homds.nphase:end), cell2mat(bif_points(1,2:end-
↪1))), ...
      homoclinic_br1(homds.coords(3:homds.nphase:end), cell2mat(bif_points(1,2:end-
↪1))), ...
      'Color', cm(2,:), 'HandleVisibility', 'Off', 'LineWidth', 2)
xlabel('$g_1$')
```

(continues on next page)

(continued from previous page)

```
ylabel('$g_2$')
zlabel('$g_3$')
plot3(bt1.x(1), bt1.x(2), bt1.x(3), '.k' ,'MarkerSize', 20)
legend('Bogdanov-Takens point', 'Location', 'SouthEast')
title('Homoclic orbits in $(g_1,g_2,g_3)$-phase space')
grid on
view(66, 50)
```



19.14.1 Predictors of orbits for various epsilons

Before proceeding with continuing the homoclinic orbits emanating from the remaining three Bogdanov-Takens points we show that the estimate of the amplitude is very conservative. Below we compute for a large range of amplitudes the predicted and corrected homoclinic solutions and compare them in phase space. We see that for amplitudes up to `1.0e-02` the predicted homoclinic orbits are indistinguishable.

```
options = BT_Hom_set_options();
options.messages = false;
options.correct = false;
options.TTolerance = 1.0e-05;

amplitudes = linspace(1.0e-03, 5.0e-02, 10);
```

(continues on next page)

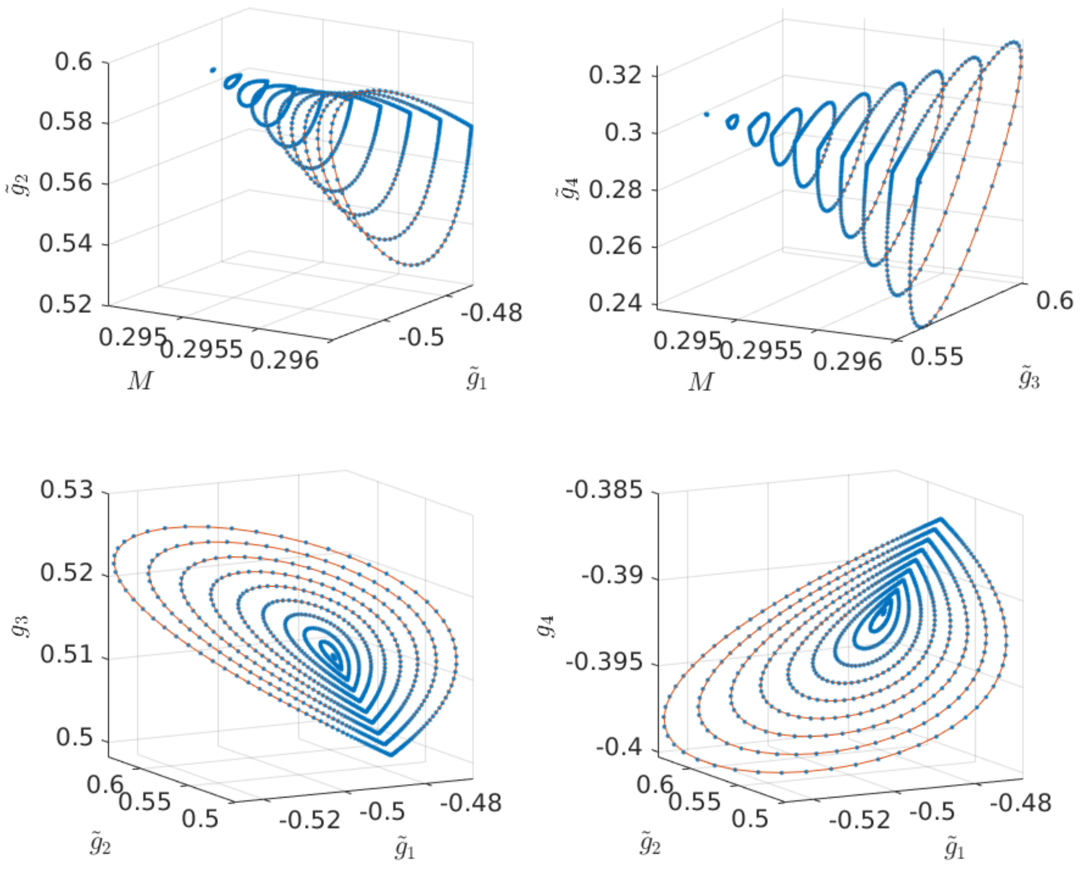
(continued from previous page)

```
XPredicted = zeros(660,length(amplitudes));
XCorrected = zeros(660,length(amplitudes));
for j=1:length(amplitudes)
  options.amplitude = amplitudes(j);
  [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, options);
  XPredicted(:,j) = x_pred;
  try
    XCorrected(:,j) = newtcorr(x_pred, v0);
  catch
    warning('Didn''t convergence to homoclinic solution')
  end
end

clf
subplot(2,2,1); hold on
R = @(alpha) [cos(alpha) -sin(alpha); sin(alpha) cos(alpha)];
S = @(s) [s 0; 0 1];
parsCorrected = XCorrected(homds.PeriodIdx+1,1:end).*ones(homds.tps,10);
for i=1:length(amplitudes)
    [profile, saddle] = bt_rearr(XCorrected(:,i));
    profile = reshape(profile,4,[]);
    profileRotated = R(-1.2181)*(S(100)*R(1.2181)*(profile(1:2,:) ...
                        - saddle(1:2)) + saddle(1:2));
    plot3(parsCorrected(:,i),profileRotated(1,:)', profileRotated(2,:)', ...
          '.','color', cm(1,:))

    [profile, saddle] = bt_rearr(XPredicted(:,i));
    profile = reshape(profile,4,[]);
    profileRotated = R(-1.2181)*(S(100)*R(1.2181)*(profile(1:2,:) ...
                        - saddle(1:2)) + saddle(1:2));
    plot3(parsCorrected(:,i),profileRotated(1,:)', profileRotated(2,:)', ...
          'color', cm(2,:))
end
xlabel('$M$')
ylabel('$\tilde g_1$')
zlabel('$\tilde g_2$')
grid on
view(32,15)

subplot(2,2,2); hold on
for i=1:length(amplitudes)
    alpha0 = -1.139;
    [profile, saddle] = bt_rearr(XCorrected(:,i));
    profile = reshape(profile,4,[]);
    profileRotated = R(-alpha0)*(S(100)*R(alpha0)*(profile(3:4,:) ...
                        - saddle(3:4)) + saddle(3:4));
    plot3(parsCorrected(:,i),profileRotated(1,:)', profileRotated(2,:)', ...
          '.','color', cm(1,:))

    [profile, saddle] = bt_rearr(XPredicted(:,i));
    profile = reshape(profile,4,[]);
    profileRotated = R(-alpha0)*(S(100)*R(alpha0)*(profile(3:4,:) ...
                        - saddle(3:4)) + saddle(3:4));
    plot3(parsCorrected(:,i),profileRotated(1,:)', profileRotated(2,:)', ...
          'color', cm(2,:))
end
xlabel('$M$')
```

(continues on next page)

(continued from previous page)

```
ylabel('$\tilde g_3$')
zlabel('$\tilde g_4$')
grid on
view(28,15)

subplot(2,2,3); hold on
R = @(alpha) [cos(alpha) -sin(alpha) 0; sin(alpha) cos(alpha) 0; 0 0 1];
S = @(s) [s 0 0; 0 1 0; 0 0 1];
for i=1:length(amplitudes)
    [profile, saddle] = bt_rearr(XCorrected(:,i));
    profile = reshape(profile,4,[]);
    profileRotated = R(-1.2181)*(S(200)*R(1.2181)*(profile(1:3,:) ...
                        - saddle(1:3)) + saddle(1:3));
    plot3(profileRotated(1,:)', profileRotated(2,:)', profileRotated(3,:)', ...
          '.','color', cm(1,:))

    [profile, saddle] = bt_rearr(XPredicted(:,i));
    profile = reshape(profile,4,[]);
    profileRotated = R(-1.2181)*(S(200)*R(1.2181)*(profile(1:3,:) ...
                        - saddle(1:3)) + saddle(1:3));
    plot3(profileRotated(1,:)', profileRotated(2,:)', profileRotated(3,:)', ...
          'color', cm(2,:))
end
xlabel('$\tilde g_1$')
ylabel('$\tilde g_2$')
zlabel('$g_3$')
grid on
view(332,11)

subplot(2,2,4); hold on
for i=1:length(amplitudes)
    [profile, saddle] = bt_rearr(XCorrected(:,i));
    profile = reshape(profile,4,[]);
    profileRotated = R(-1.2181)*(S(200)*R(1.2181)*(profile([1,2,4],:) ...
                        - saddle([1,2,4])) + saddle([1,2,4]));
    plot3(profileRotated(1,:)', profileRotated(2,:)', profileRotated(3,:)', ...
          '.','color', cm(1,:))

    [profile, saddle] = bt_rearr(XPredicted(:,i));
    profile = reshape(profile,4,[]);
    profileRotated = R(-1.2181)*(S(200)*R(1.2181)*(profile([1,2,4],:) ...
                        - saddle([1,2,4])) + saddle([1,2,4]));
    plot3(profileRotated(1,:)', profileRotated(2,:)', profileRotated(3,:)', ...
          'color', cm(2,:))
end
xlabel('$\tilde g_1$')
ylabel('$\tilde g_2$')
zlabel('$g_4$')
grid on
view(332,11)
```



19.15 Continue limit points emanating from the Bogdanov-Takens point

Next we also continue the limit points emanating from the Bogdanov-Takens points.

```
[lp1_x, lp1_v] = init_BT_LP(odefile, bt1.x, bt1.par, ap);
[lp_br, ~, lp_br1_bif] = cont(@limitpoint, lp1_x, lp1_v, opt);
opt.Backward = 1;
lp_br_rev = cont(@limitpoint, lp1_x, lp1_v, opt);
```

```
first point found
tangent vector to first point found
label = BT, x = ( -0.133317 0.022458 0.090992 -0.220100 0.883754 4.419949 )
(a,b)=(-2.411266e-02, -5.207995e-01)
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.
 RCOND =  1.198603e-16.
> In limitpoint>curve_func (line 26)
In newtcorr (line 21)
In cont (line 358)
```

(continues on next page)

(continued from previous page)

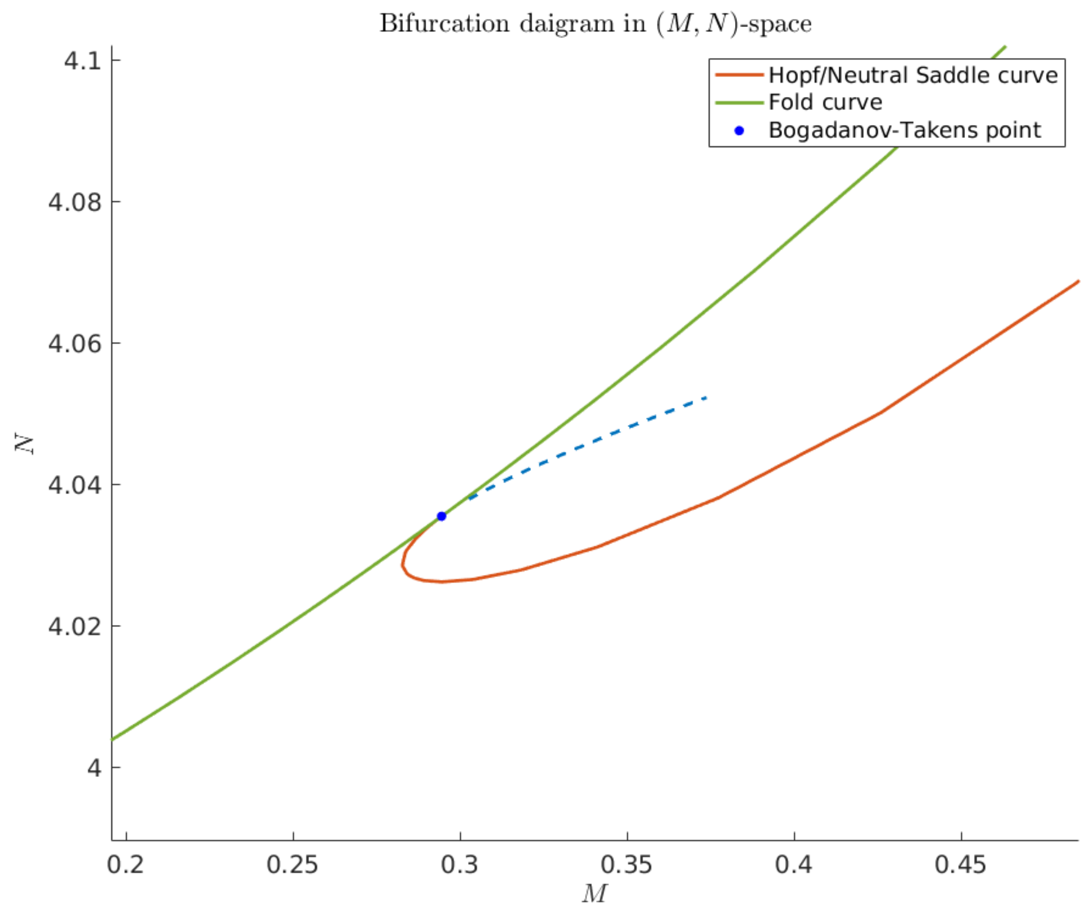
```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
 ↪RCOND = 4.557903e-114.
> In limitpoint>curve_func (line 26)
In newtcorr (line 21)
In cont (line 358)

Warning: Matrix is singular, close to singular or badly scaled. Results may be␣
 ↪inaccurate. RCOND = NaN.
> In limitpoint>curve_func (line 26)
In newtcorr (line 21)
In cont (line 358)

label = CP, x = ( -0.068275 0.012534 0.046191 -0.194560 0.905065 4.487600 )
c=1.231052e+00
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
 ↪RCOND =  2.445239e-19.
> In limitpoint>curve_func (line 26)
In newtcorr (line 21)
In cont>LocateTestFunction (line 945)
In cont (line 465)

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.␣
 ↪RCOND = 3.462497e-125.
> In limitpoint>curve_func (line 26)
In newtcorr (line 21)
In cont>LocateTestFunction (line 945)
In cont (line 465)

Warning: Matrix is singular, close to singular or badly scaled. Results may be␣
 ↪inaccurate. RCOND = NaN.
> In limitpoint>curve_func (line 26)
In newtcorr (line 21)
In cont>LocateTestFunction (line 945)
In cont (line 465)

label = ZH, x = ( 0.089558 0.013324 -0.045799 -0.157071 1.383946 3.902668 )
Neutral saddle
Zero-Neutral Saddle

elapsed time  = 0.3 secs
npoints curve = 50
first point found
tangent vector to first point found
label = BT, x = ( -0.715157 -0.250968 0.510051 -0.391935 0.294477 4.035536 )
(a,b)=(-1.387256e+00, 1.894680e-01)

elapsed time  = 0.1 secs
npoints curve = 50
```

We see that there are two additional Bogdanov-Takens points detected. We extract these below.

19.16 Bifurcation plot

Next we plot the continued curves in (M, N) parameter space near the first Bogdanov-Takens point.

```
%plot inline
hold on
homColor  = cm(1,:);
hopfColor = cm(2,:);
foldColor = cm(5,:);
plot(hopf_br(5,:), hopf_br(6,:), 'Color', hopfColor, 'linewidth', 2)
plot(lp_br(5,:), lp_br(6,:), 'Color', foldColor, 'linewidth', 2)
plot(lp_br_rev(5,:), lp_br_rev(6,:), 'Color', foldColor, 'linewidth', 2, ...
    'HandleVisibility', 'Off', 'linewidth', 2);
plot(BTPoint1(5), BTPoint1(6), '.b' ,'MarkerSize', 20)
plot(homoclinic_br1(homds.PeriodIdx+1,:), ...
     homoclinic_br1(homds.PeriodIdx+2,:), ...
     '--', 'Color', homColor, 'linewidth', 2, 'HandleVisibility', 'Off')
xlabel('$M$')
ylabel('$N$')
legend({'Hopf/Neutral Saddle curve', 'Fold curve',...
    'Bogadanov-Takens point'}, 'Location', 'NorthEast')
title('Bifurcation daigram in $(M,N)$-space')
axis([0.1956    0.4852    3.9896    4.1020])
```



19.17 Convergence plot

We finish this notebook with a log-log convergence plot comparing the different third order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the first Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`).

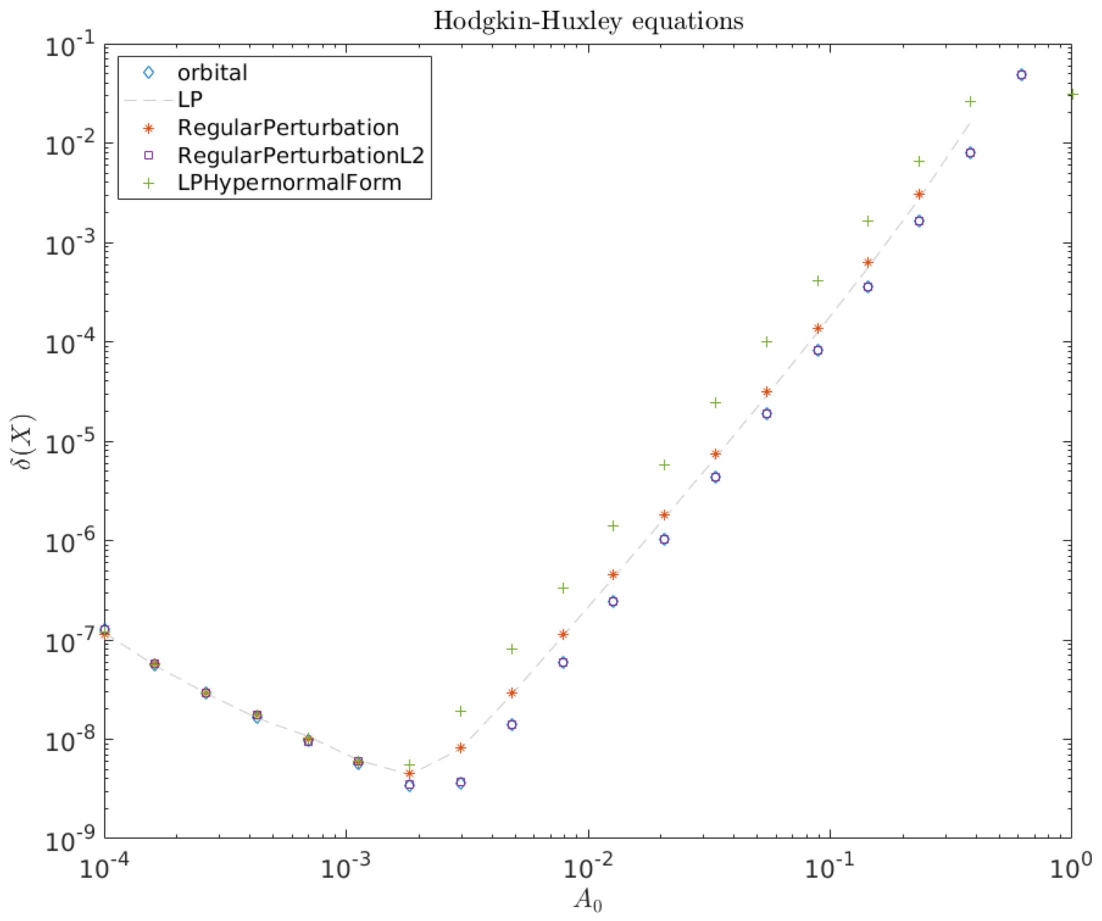
```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 0, 20);
methodList = {'orbital', 'LP', 'RegularPerturbation', ...
    'RegularPerturbationL2', 'LPHypernormalForm'};
relativeErrors = {};
for i=1:length(methodList)
    for o=1:3
        BToptions.method = methodList{i};
        BToptions.order = o;
        relativeErrors{o,i} = zeros(size(amplitudes));
        for j=1:length(amplitudes)
            BToptions.amplitude = amplitudes(j);
            [x_pred, v0] = init_BT_Hom(odefile, bt1, ap, BToptions);
            try
                x_corrected = newtcorr(x_pred, v0);
                relativeErrors{o,i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
            catch
                warning('Did not converge.')
                continue
            end
        end
    end
end

cm = lines();
loglog(amplitudes, relativeErrors{3,1}(:), 'd', ...
       amplitudes, relativeErrors{3,2}(:), '--', ...
       amplitudes, relativeErrors{3,3}(:), '*', ...
       amplitudes, relativeErrors{3,4}(:), 's', ...
       amplitudes, relativeErrors{3,5}(:), '+')
legend(methodList, 'Location', 'NorthWest')
title('Hodgkin-Huxley equations')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); [0.8 0.8 0.8]; cm(2,:); cm(4,:); cm(5,:)];
```

```
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
```

(continues on next page)

(continued from previous page)

```
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
Warning: Did not converge.
```



CHAPTER

TWENTY

SAVE DATA TO FILES

```
writematrix([amplitudes', relativeErrors{1,3}(:)], '../../data/HomRGflowsRPorder1.csv
↪', 'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{2,3}(:)], '../../data/HomRGflowsRPorder2.csv
↪', 'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{3,3}(:)], '../../data/HomRGflowsRPorder3.csv
↪', 'Delimiter', ' ')

writematrix([amplitudes', relativeErrors{1,2}(:)], '../../data/HomRGflowsLPorder1.csv
↪', 'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{2,2}(:)], '../../data/HomRGflowsLPorder2.csv
↪', 'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{3,2}(:)], '../../data/HomRGflowsLPorder3.csv
↪', 'Delimiter', ' ')

writematrix([amplitudes', relativeErrors{3,1}(:)], '../../data/
↪HomRGflowsLPorder3orbital.csv', 'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{3,4}(:)], '../../data/
↪HomRGflowsRegularPerturbationL2.csv', 'Delimiter', ' ')
writematrix([amplitudes', relativeErrors{3,5}(:)], '../../data/
↪HomRGflowsLPHypernormalForm.csv', 'Delimiter', ' ')
```

Part X

Bogdanov-Takens

CHAPTER

TWENTYONE

GENERATE SYSTEM FILES FOR PREDATOR-PREY SYSTEM WITH CONSTANT RATE HARVESTING

In this script the **system files** for critical Bogdanov-Takens codimension two normal form given by

$$\begin{cases} \dot{w}_0 = w_1, \\ \dot{w}_1 = \beta_1 + \beta_2 w_1 - w_0^2 + w_0 w_1. \end{cases}$$

are generated. These are used in the *BogdanovTakens.md* demo.

21.1 Add MatCont path and load sym package if GNU Octave is used

```
matcontpath = '../';
addpath(matcontpath);
addpath([matcontpath, '/Utilities']);
if isOctave
  pkg load symbolic % for GNU Octave
end
```

21.2 Set the system name

```
system_name = 'BogdanovTakensNormalForm';
```

21.3 Create coordinates and parameter names as strings

```
coordsnames = {'w0', 'w1'};
parnames = {'beta1', 'beta2'};
```

21.4 Create symbols for coordinates and parameters

The array `par` is the array of symbols in the same order as parnames. Due to the following two lines we may, for example, use either `alpha` or `par(1)`. There should no changes be need of this code.

```
syms(parnames{:});        % create symbol for alpha and delta
par=cell2sym(parnames);   % now alpha1 is par(1) etc
syms(coordsnames{:});     % create symbol for alpha and delta
coords=cell2sym(coordsnames); % create 1 x n vector for coordinates
```

21.5 Define the system

```
dw0_dt = w1;
dw1_dt = beta1 + beta2*w1 - w0^2 + w0*w1;
system = [dw0_dt; dw1_dt];
```

In general there are no modifications needed after this line.

21.6 Differentiate and generate code (directional derivatives)

Exporting it to `<system_name>.m`. This method uses directional derivatives. Then using polarization identities derivatives can be calculated in arbitrary direction.

```
suc = generate_directional_derivatives(...
  system,...    % n x 1 array of derivative symbolic expressions
  coords,... % 1 x n array of symbols for states
  par,...       % 1 x np array of symbols used for parameters
  system_name,... % argument specifying the system name
  [matcontpath, 'Systems/']... % directory to save to file
);
```

21.7 Higher-order parameter-dependent multi-linear form.

Exporting it to `<system_name>_multilinearforms.m`. These multi-linear forms are currently only used in the computation of the parameter-dependent center manifold for the codimension two Bogdanov-Takens bifurcation.

```
order = 3;
suc = generate_multilinear_forms(system_name, system, coords, par, order, ...
        [matcontpath, 'Systems/']);
```

CHAPTER

TWENTYTWO

BOGDANOV-TAKENS NORMAL FORM

The universal unfolding for the Bogdanov-Takens normal form is given by

$$\begin{cases} \dot{w}_0 = w_1, \\ \dot{w}_1 = \beta_1 + \beta_2 w_1 + w_0^2 - w_0 w_1. \end{cases} \tag{1}$$

22.1 Overview

In this demo we will

- Compare the profiles of the predicted homoclinic solutions emanating from the Bogdanov-Takens point with the corrected homoclinic solutions curve in phase-space. We will do this using the asymptotics derived in [Kuz21] and the asymptotics as provided in [AHGKM16]. We will focus on the difference in using the higher order approximation of the non-linear time transformation.
- Create a convergence plot of the different approximation derived in [Kuz21]. And compare them with the approximation derived in [AHGKM16].

This demo will not show how to continue the homoclinic curve emanating from the codimension two Bogdanov-Takens point. For this we refer to one of the other demos.

22.2 Load MatCont

Before we can start using **MatCont** we need to add the main directory of **MatCont,** as well as various subdirectories of **MatCont,** to the *MATLAB search path*. This is done in the code below. The variable `matcont_home` should point to the main directory of **MatCont.**

```
clear all
matcontpath = '../';
addpath(matcontpath)
addpath([matcontpath, 'Systems'])
addpath([matcontpath, 'Equilibrium'])
addpath([matcontpath, 'LimitPoint'])
addpath([matcontpath, 'LimitPointCycle'])
addpath([matcontpath, 'Hopf'])
addpath([matcontpath, 'Homoclinic'])
addpath([matcontpath, 'LimitCycle'])
addpath([matcontpath, 'Continuer'])
addpath([matcontpath, 'MultilinearForms'])
addpath([matcontpath, 'Utilities'])
set(groot, 'defaultTextInterpreter', 'LaTeX');
set(0,'defaultAxesFontSize',15)
```

22.3 Set the odefile

Next we set the variable `odefile` to the *system file* previously generated by the notebook *Generate system files for predator-prey system with constant rate harvesting*.

```
odefile=@BogdanovTakensNormalForm;
```

22.4 Define Bogdanov-Takens point

```
w0 = 0;
w1 = 0;
beta1 = 0;
beta2 = 0;
bt.x = [w0; w1];
bt.par = [beta1; beta2];
```

To refer to the parameters throughout the script we create a **cell array** of strings containing the parameter names. This is then converted into a **struct**. This allows us to refer to the parameters as `ind.parametername`, similar as done in *DDE-BifTool*.

```
parnames = {'beta1', 'beta2'};
cind = [parnames;num2cell(1:length(parnames))];
ind  = struct(cind{:});
```

22.5 Lindstedt-Poincaré with and without higher order time approximation

Here we will show that it is essential to use the higher order approximation of the non-linear time transformation as derived in [Kuz21]. This was not taken into account in [AHGKM16].

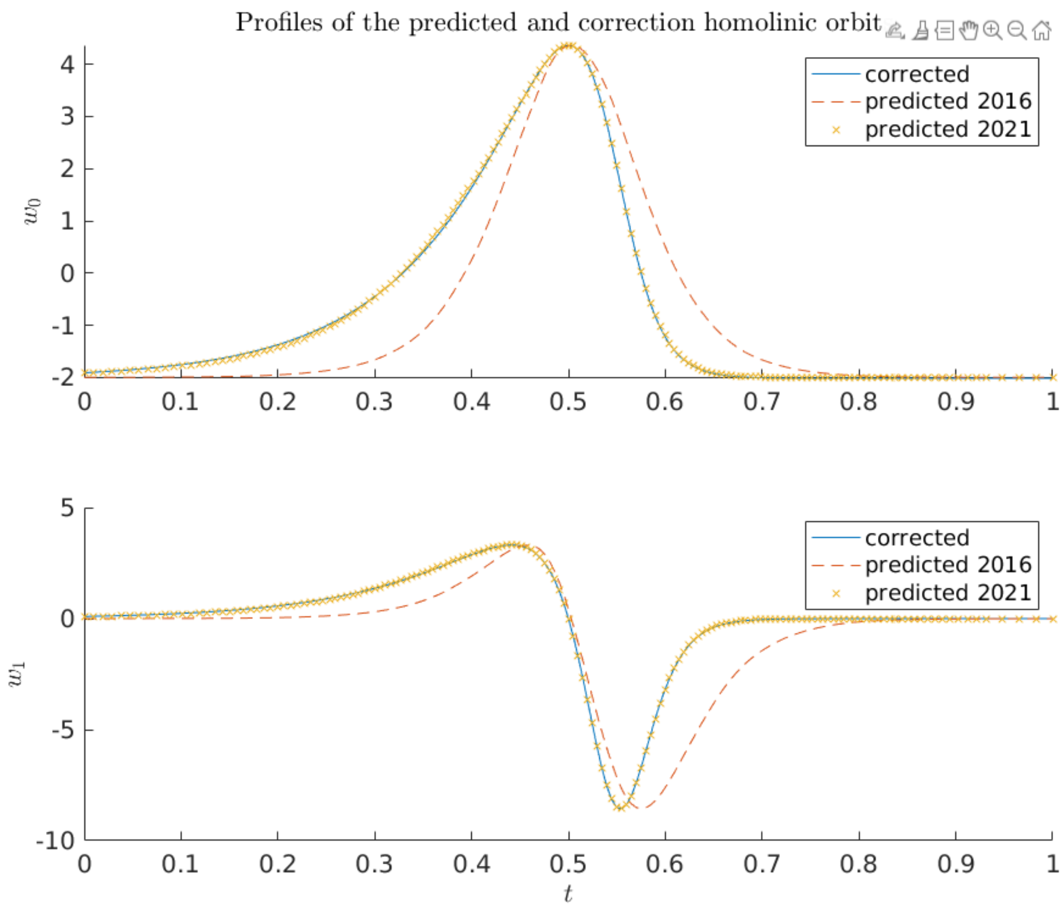
```
%plot --width 1024 --height 800
ap = [ind.beta1, ind.beta2];
BToptions = BT_Hom_set_options();
BToptions.method = 'LP';
BToptions.HigherOrderTimeReparametrization = 0;
BToptions.correct = 0;
BToptions.amplitude = 6;
BToptions.TTolerance = 1.0e-03;
hom2016pred = init_BT_Hom(odefile, bt,  ap, BToptions);
BToptions.HigherOrderTimeReparametrization = 1;
[hom2021pred, hom_v_pred] = init_BT_Hom(odefile, bt,  ap, BToptions);
homcorrected = newtcorr(hom2021pred, hom_v_pred);
global homds
[hom2016predOrbit, saddle2016pred] = bt_rearr(hom2016pred);
[hom2021predOrbit, saddle2021pred] = bt_rearr(hom2021pred);
[homcorrectedOrbit, saddlecorrected] = bt_rearr(homcorrected);
subplot(2,1,1); hold on;
title('Profiles of the predicted and correction homolinic orbits.')
plot(homds.finemsh, homcorrectedOrbit(1:2:end),'-')
plot(homds.finemsh, hom2016predOrbit(1:2:end),'--')
```

(continues on next page)

(continued from previous page)

```
plot(homds.finemsh, hom2021predOrbit(1:2:end),'x')
legend({'corrected', 'predicted 2016', 'predicted 2021'})
ylabel('$w_0$')
subplot(2,1,2); hold on;
plot(homds.finemsh, homcorrectedOrbit(2:2:end),'-')
plot(homds.finemsh, hom2016predOrbit(2:2:end),'--')
plot(homds.finemsh, hom2021predOrbit(2:2:end),'x')
legend({'corrected', 'predicted 2016', 'predicted 2021'})
ylabel('$w_1$')
xlabel('$t$')
```

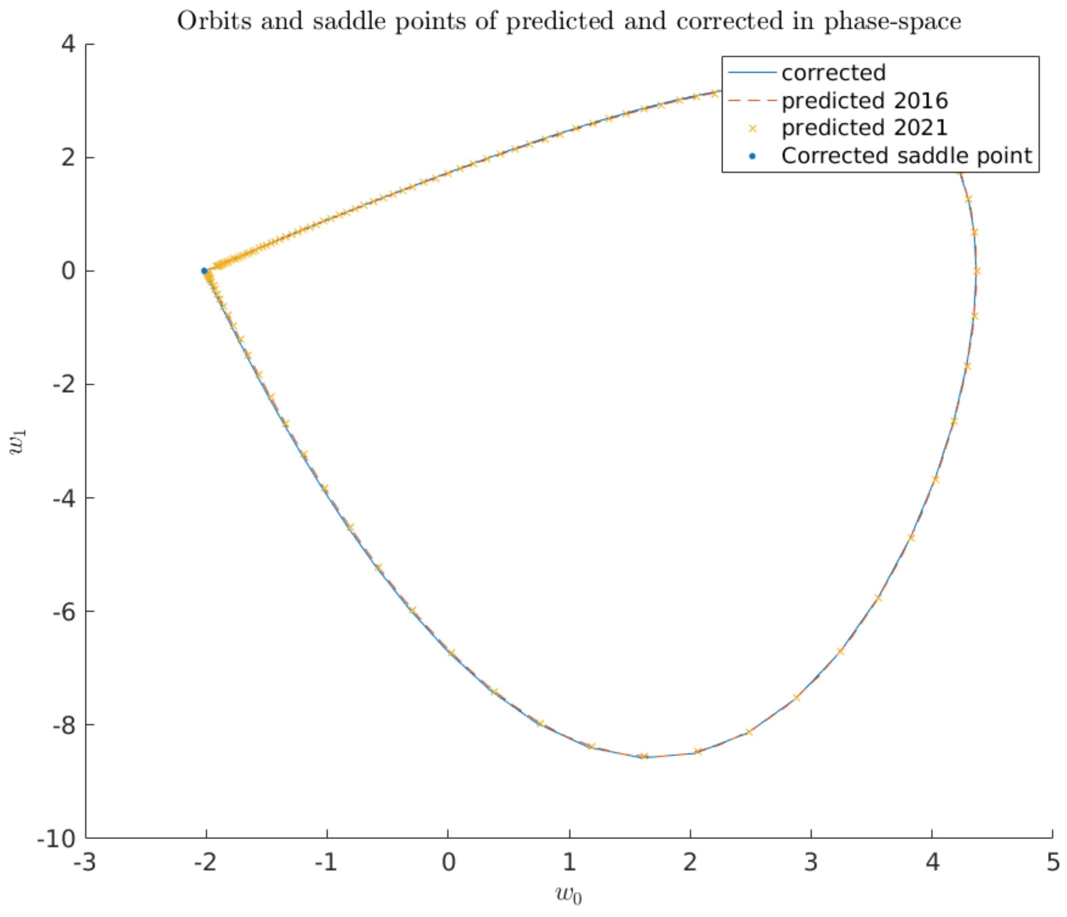
```
BT normal form coefficients:
a=-1,          b=1
T: 5.04286
The initial perturbation parameter:  1
The initial distance eps0: 0.00144308
The initial distance eps1: 0.00485318
BT normal form coefficients:
a=-1,          b=1
T: 5.04286
The initial perturbation parameter:  1
The initial distance eps0: 0.112165
The initial distance eps1: 4.58758e-08
```



22.6 Compare predictor and corrected solution in (w_0, w_1) phase-space

The plot below shows that both predicted homoclinic orbits are in excellent agreement in (w_0, w_1) phase-space. However, this is not on which the Newton iteration operates. It operates on the profiles as shown above.

```
hold on
plot(homcorrectedOrbit(1:2:end),homcorrectedOrbit(2:2:end), '-')
plot(hom2016predOrbit(1:2:end),hom2016predOrbit(2:2:end),'--')
plot(hom2021predOrbit(1:2:end),hom2021predOrbit(2:2:end),'x')
plot(saddlecorrected(1), saddlecorrected(2), '.', 'MarkerSize', 12, 'Color', [0 0.
↪4470 0.7410])
xlabel('$w_0$')
ylabel('$w_1$')
legend({'corrected', 'predicted 2016', 'predicted 2021', 'Corrected saddle point'})
title('Orbits and saddle points of predicted and corrected in phase-space')
```



22.7 Convergence plots

Below we show a log-log convergence plot comparing the different higher order homoclinic approximation methods derived in [Kuz21] to approximate the homoclinic solutions near the Bogdanov-Takens point. On the abscissa is the amplitude A_0 and on the ordinate the relative error δ between the constructed solution (`x_pred`) to the defining system for the homoclinic orbit and the Newton corrected solution (`x_corrected`). We will also plot the third order predictor as derived in [AHGKM16]. It is shown that it only provides zeroth-order accuracy.

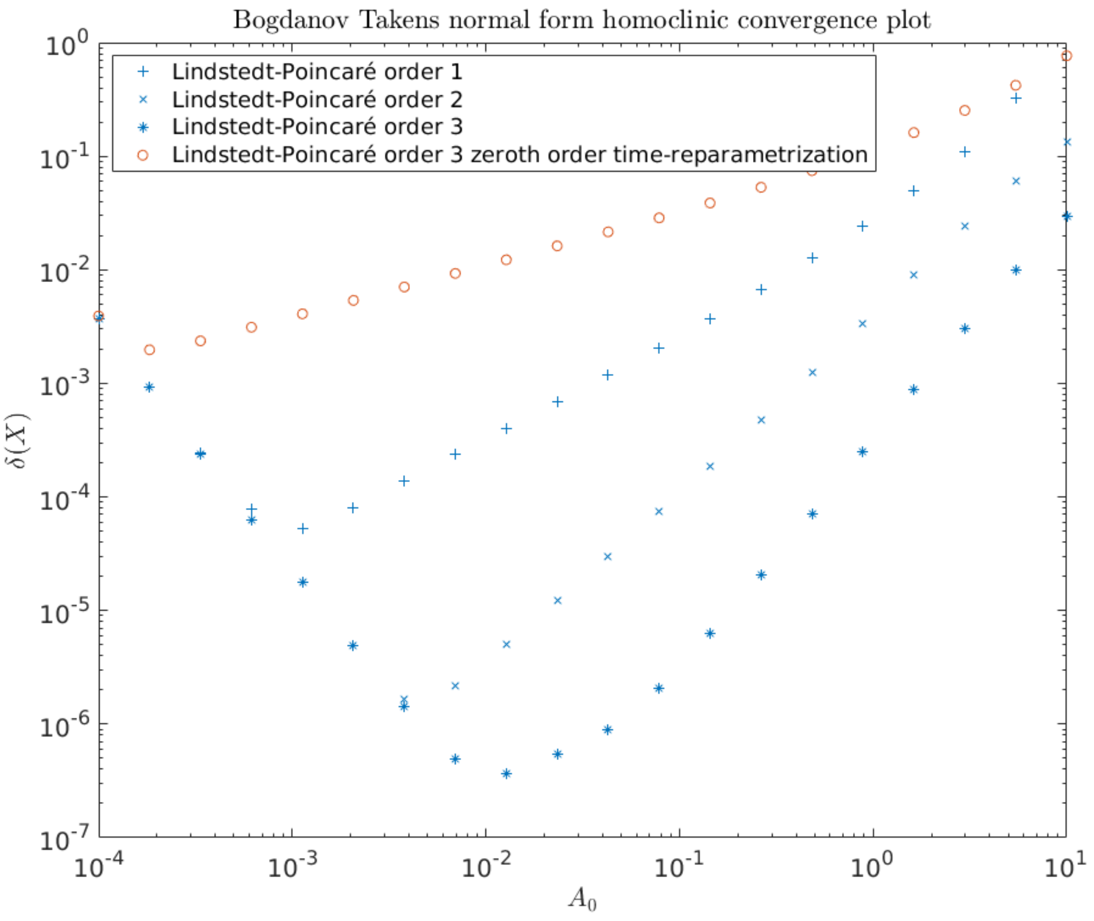
```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 1, 20);

BToptions.method = 'LP';
relativeErrors = {};
for i=1:3
    relativeErrors{i} = zeros(size(amplitudes));
    BToptions.order = i;
    for j=1:length(amplitudes)
        BToptions.amplitude = amplitudes(j);
        [x_pred, v0] = init_BT_Hom(odefile, bt, ap, BToptions);
        try
            x_corrected = newtcorr(x_pred, v0);
            relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
        catch
            warning('Did not converge.')
            continue
        end
  end
end

i=4;
relativeErrors{i} = zeros(size(amplitudes));
BToptions.HigherOrderTimeReparametrization = 0;
for j=1:length(amplitudes)
    BToptions.amplitude = amplitudes(j);
    [x_pred, v0] = init_BT_Hom(odefile, bt, ap, BToptions);
    try
        x_corrected = newtcorr(x_pred, v0);
        relativeErrors{i}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
    catch
        warning('Did not converge.')
        continue
    end
end

cm = lines();
loglog(amplitudes, relativeErrors{1}(:), '+', ...
       amplitudes, relativeErrors{2}(:), 'x', ...
       amplitudes, relativeErrors{3}(:), '*', ...
       amplitudes, relativeErrors{4}(:), 'o')
legend('Lindstedt-Poincaré order 1', 'Lindstedt-Poincaré order 2', ...
'Lindstedt-Poincaré order 3', ...
'Lindstedt-Poincaré order 3 zeroth order time-reparametrization', 'Location',
 ↪'NorthWest')
```

(continues on next page)

(continued from previous page)

```
title('Bogdanov Takens normal form homoclinic convergence plot')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
ax.ColorOrder = [cm(1,:); cm(1,:); cm(1,:); cm(2,:); cm(5,:)];
```

```
Warning: Did not converge.
```



We finish this notebook with a convergence plot as above. This time we show that the $L2$ phase-condition provides better accuracy then the phase-condition used in [AHGKM16]. However, as stated in [Kuz21] these results do not lift to the central manifold of a general system.

Also shown is that the Lindstedt-Poincaré method provides a much better approximation here then either one of the regular perturbation methods. This however doens't hold true for the first order correction.

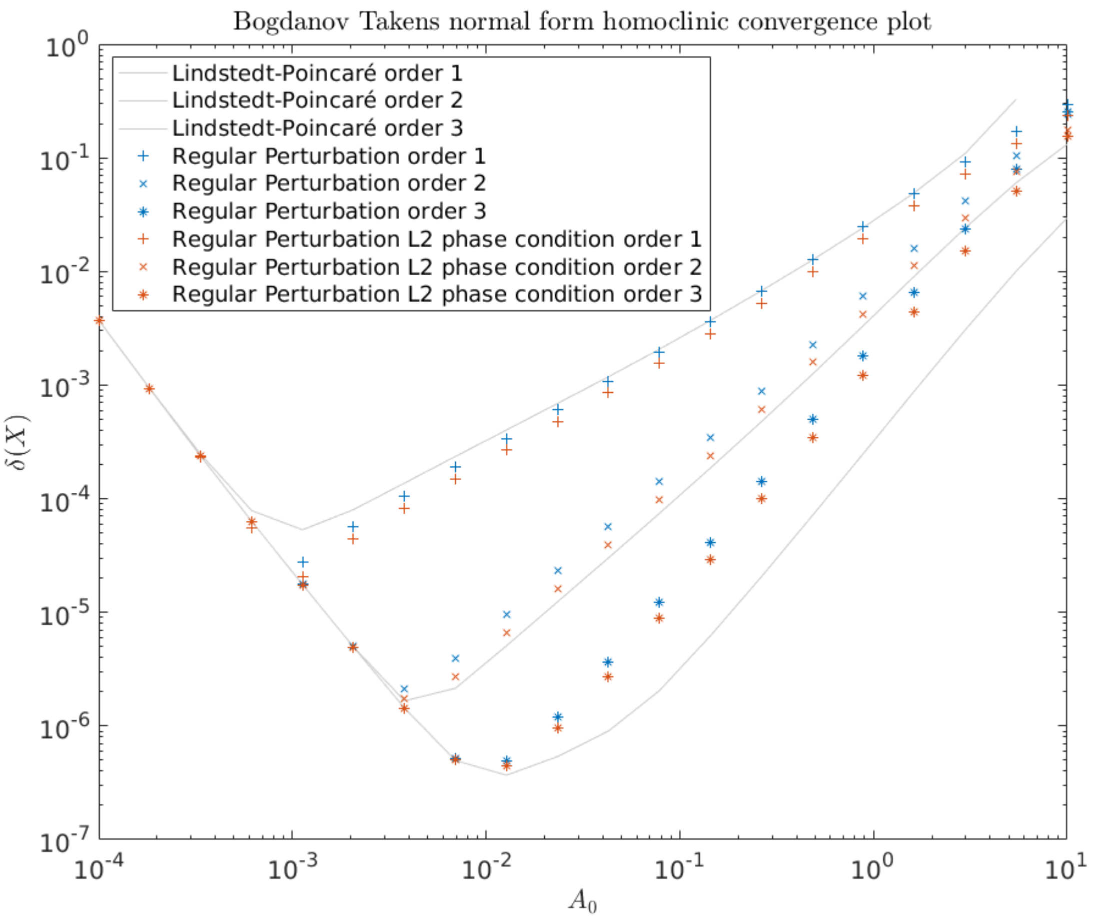
```
BToptions = BT_Hom_set_options();
BToptions.TTolerance = 1e-05;
BToptions.messages = false;
BToptions.correct = false;

amplitudes = logspace(-4, 1, 20);
methodList = {'LP', 'RegularPerturbation', 'RegularPerturbationL2'};
```

(continues on next page)

(continued from previous page)

```
relativeErrors = {};
for i=1:3
    for k=1:length(methodList)
        relativeErrors{i,k} = zeros(size(amplitudes));
        BToptions.order = i;
        BToptions.method = methodList{k};
        for j=1:length(amplitudes)
            BToptions.amplitude = amplitudes(j);
            [x_pred, v0] = init_BT_Hom(odefile, bt, ap, BToptions);
            try
                x_corrected = newtcorr(x_pred, v0);
                relativeErrors{i,k}(j) = norm(x_corrected-x_pred)/norm(x_corrected);
            catch
                warning('Did not converge.')
                continue
            end
        end
    end
end

clf
cm = lines();
loglog(amplitudes, relativeErrors{1,1}(:), '-', ...
       amplitudes, relativeErrors{2,1}(:), '-', ...
       amplitudes, relativeErrors{3,1}(:), '-', ...
       amplitudes, relativeErrors{1,2}(:), '+', ...
       amplitudes, relativeErrors{2,2}(:), 'x', ...
       amplitudes, relativeErrors{3,2}(:), '*', ...
       amplitudes, relativeErrors{1,3}(:), '+', ...
       amplitudes, relativeErrors{2,3}(:), 'x', ...
       amplitudes, relativeErrors{3,3}(:), '*')
legend('Lindstedt-Poincaré order 1', 'Lindstedt-Poincaré order 2', ...
'Lindstedt-Poincaré order 3', ...
'Regular Perturbation order 1','Regular Perturbation order 2', ...
'Regular Perturbation order 3', ...
'Regular Perturbation L2 phase condition order 1', ...
'Regular Perturbation L2 phase condition order 2', ...
'Regular Perturbation L2 phase condition order 3', ...
'Location', 'NorthWest')
title('Bogdanov Takens normal form homoclinic convergence plot')
xlabel('$A_0$')
ylabel('$\delta(X)$')
ax = gca;
gray = [0.8 0.8 0.8];
ax.ColorOrder = [gray; gray; gray; cm(1,:); cm(1,:); cm(1,:); ...
                 cm(2,:); cm(2,:); cm(2,:)];
```

```
Warning: Did not converge.
```


Bogdanov Takens normal form homoclinic convergence plot
Lindstedt-Poincaré order 1
Lindstedt-Poincaré order 2
Lindstedt-Poincaré order 3
Regular Perturbation order 1
Regular Perturbation order 2
Regular Perturbation order 3
Regular Perturbation L2 phase condition order 1
Regular Perturbation L2 phase condition order 2
Regular Perturbation L2 phase condition order 3
$\delta(X)$
A_0
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}

Part XI

References

CHAPTER

TWENTYTHREE

REFERENCES

BIBLIOGRAPHY

[AHGKM16] B. Al-Hdaibat, W. Govaerts, Yu. A. Kuznetsov, and H. G. E. Meijer. Initialization of homoclinic solutions near bogdanov–takens points: lindstedt–poincaré compared with regular perturbation method. *SIAM Journal on Applied Dynamical Systems*, 15(2):952–980, 2016. arXiv:https://doi.org/10.1137/15M1017491, doi:10.1137/15M1017491.

[BAH15] Willy Govaerts Bashir Al-Hdaibat. Continuation of homoclinic orbits starting from a generic bogdanov-takens point with matcont. *SourceForge*, 2015. URL: https://sourceforge.net/projects/matcont/files/NewestDocumentation/MatContODE/AdvancedTutorials/labinithom2.pdf.

[BR00] A.S. Bazanella and R. Reginatto. Robustness margins for indirect field-oriented control of induction motors. *IEEE Trans. Automat. Contr.*, 45(6):1226–1231, June 2000. doi:10.1109/9.863613.

[Baz85] A. D. Bazykin. \cyr *Matematicheskaya biofizika vzaimodeĭ* \cyr *stvuyushchikh populyatsiĭ*. "Nauka", Moscow, 1985.

[Bey94] W.-J. Beyn. Numerical analysis of homoclinic orbits emanating from a takens-bogdanov point. *IMA J Numer Anal*, 14(3):381–410, 1994. doi:10.1093/imanum/14.3.381.

[BS79] F. Brauer and A.C. Soudack. Stability regions and transition phenomena for harvested predator-prey systems. *J. Math. Biol.*, 7(4):319–337, 1979. doi:10.1007/bf00275152.

[BIK78] V.I. Bykov, G.S. Iablonskii, and V.F. Kim. Simple-model of kinetic auto-oscillation model in catalytic reaction of co-oxidation. *Doklady Akademii Nauk SSSR*, 242:637–639, 1978.

[DGK+08] A. Dhooge, W. Govaerts, Yu.A. Kuznetsov, H.G.E. Meijer, and B. Sautois. New features of the software matcontfor bifurcation analysis of dynamical systems. *Mathematical and Computer Modelling of Dynamical Systems*, 14(2):147–175, April 2008. doi:10.1080/13873950701742754.

[HH52a] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952. arXiv:https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1952.sp004764, doi:https://doi.org/10.1113/jphysiol.1952.sp004764.

[HH52b] A. L. Hodgkin and A. F. Huxley. Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *The Journal of Physiology*, 116(4):449–472, 1952. arXiv:https://physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1952.sp004717, doi:https://doi.org/10.1113/jphysiol.1952.sp004717.

[JP21] C. B. Jepsen and F. K. Popov. Homoclinic rg flows, or when relevant operators become irrelevant. *arXiv*, 2021. URL: arXiv:2105.01625v1.

[Khi90] A. I. Khibnik. *LINLBF: A Program for Continuation and Bifurcation Analysis of Equilibria Up to Codimension Three*, pages 283–296. Springer Netherlands, Dordrecht, 1990. doi:10.1007/978-94-009-0659-4_19.

[Kuz21] Bosschaert M.M. Kuznetsov, I.A. Interplay between normal forms and center manifold reduction for homoclinic predictors near bogdanov-takens bifurcation. *arXiv*, 2021.

[KMGS08] Yu.A. Kuznetsov, H.G.E. Meijer, W. Govaerts, and B. Sautois. Switching to nonhyperbolic cycles from codim 2 bifurcations of equilibria in odes. *Physica D: Nonlinear Phenomena*, 237(23):3061–3068, December 2008. doi:10.1016/j.physd.2008.06.006.

[Kuz04] Yu.A Kuznetsov. *Elements of Applied Bifurcation Theory*. Volume 112 of Applied Mathematical Sciences. Springer-Verlag, New York, third edition, 2004. ISBN 0-387-21906-4. doi:10.1007/978-1-4757-3978-7.

[ML81] C. Morris and H. Lecar. Voltage oscillations in the barnacle giant muscle fiber. *Biophysical Journal*, 35(1):193–213, July 1981. doi:10.1016/s0006-3495(81)84782-0.

[SGAR08] F. Salas, F. Gordillo, J. Aracil, and R. Reginatto. Codimension-two bifurcations in indirect field oriented control of induction motor drives. *Int. J. Bifurcation Chaos*, 18(03):779–792, March 2008. doi:10.1142/s0218127408020641.

[SZ14] Chunhua Shan and Huaiping Zhu. Bifurcations and complex dynamics of an sir model with the impact of the number of hospital beds. *Journal of Differential Equations*, 257(5):1662–1688, September 2014. doi:10.1016/j.jde.2014.05.030.

[ShilNikovNN95] A. Shil'Nikov, G. Nicolis, and C. Nicolis. Bifurcation and predictability analysis of a low-order atmospheric circulation model. *Int. J. Bifurcation Chaos*, 05(06):1701–1711, December 1995. doi:10.1142/s0218127495001253.

[SEL+14] Jan Sieber, Koen Engelborghs, Tatyana Luzyanina, Giovanni Samaey, and Dirk Roose. Dde-biftool manual - bifurcation analysis of delay differential equations. 2014. arXiv:arXiv:1406.7144.

[vV03] Lennaert van Veen. Baroclinic flow and the lorenz-84 model. *Int. J. Bifurcation Chaos*, 13(08):2117–2139, August 2003. doi:10.1142/s0218127403007904.

[XR99] Dongmei Xiao and Shigui Ruan. Bogdanov-Takens bifurcations in predator-prey systems with constant rate harvesting. In *Differential equations with applications to biology (Halifax, NS, 1997)*, volume 21 of Fields Inst. Commun., pages 493–506. Amer. Math. Soc., Providence, RI, 1999.