

TUTORIAL M-II:
Two-parameter bifurcation analysis of fixed points
and cycles in MatContM

April 16, 2018

MatContM is available at <http://sourceforge.net/projects/matcont/files/matcont/>. This tutorial is tested on Matlab R2017b with `matcontm5p4`. We perform a two-parameter bifurcation analysis on a discrete-time predator-prey model using MatContM.

$$\text{PPModel} : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} ax(1-x) - \frac{bxy}{1+\epsilon x} \\ \frac{dxy}{1+\epsilon x} \end{pmatrix} \quad (1)$$

The map (1) models the year-to-year dynamics of a prey-predator ecosystem. It describes how predators respond to changes in prey availability. The coordinates x and y represent the population densities of the prey and the predator, respectively, while a , b , d and ϵ are the parameters of the system. Parameters a and d determine the growth rate of the prey and predator; b determines the rate at which prey are consumed; ϵ acts as a limitation parameter on the growth of the predator population for increasing prey population density.

The fixed points of system (1) can be determined algebraically by solving the fixed point equations. One trivial solution is the origin $F_1 = (0, 0)$, which means having neither prey nor predators. For $y = 0$, we get the solution $F_2 = (\frac{a-1}{a}, 0)$, the situation where no predators exist, only prey. For $x \neq 0$ and $y \neq 0$, we get the solution

$$F_3 = \left(\frac{1}{d-\epsilon}, \frac{d}{d-\epsilon} \left(\frac{a}{b} \left(1 - \frac{1}{d-\epsilon} \right) - \frac{1}{b} \right) \right). \quad (2)$$

F_1 , F_2 and F_3 are the only fixed points of (1).

1 System specification

This map (1) is already implemented as `PredatorPreyModel` in MatContM. Select it using **System|Systembrowser** and highlighting `PredatorPreyModel`. Press **Edit** to check that the system is correct and note that all derivatives up to and including 5th order are generated symbolically. Press **OK**. Now you can actually load the system by pressing the **Load** button (see Figure 2).

Alternatively, you can implement the system yourself. Select **System|Systembrowser**, press **New** and enter the data shown in Figure 1. Here we use the name `PPModel`. In Figure 1 all derivatives up to and including 5th order are generated symbolically.

We recall that if derivatives are not provided symbolically, they will be generated by automatic differentiation with the same accuracy.

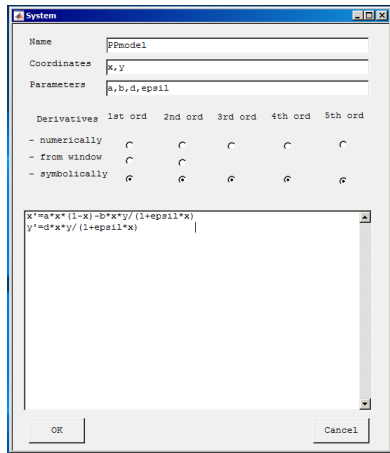


Figure 1: Entry for a new system.
Press **OK** to add the system.

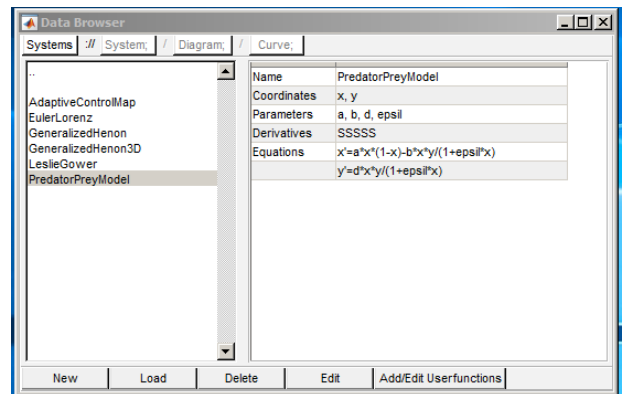


Figure 2: Press **Load** to select the high-lighted system.

2 Continuation of fixed points

We will now compute a series of fixed-point curves.

Verify that the **Initial Point Type** is set to **Fixed Point** and the **Initializer** is set to `init_FPm_FPm`. Select the **Starter** window, activate parameter **a** and input $(0,0)$ (fixed point F_1) with parameters $a = 0$, $b = 3$, $d = 3.5$ and $\text{epsil} = 1$ (Figure 3, you can set `AD threshold = 7` or keep the default 24). Open a **Plot2D** window with the layout as in Figure 4. It is also recommended to open the **Numeric** window.

Ensure that the numerical parameter values in the **Continuer** window are as follows: `InitStepSize = 0.01` while `MaxStepSize = 0.1`.

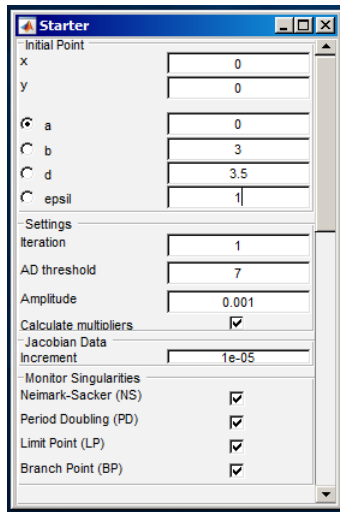


Figure 3

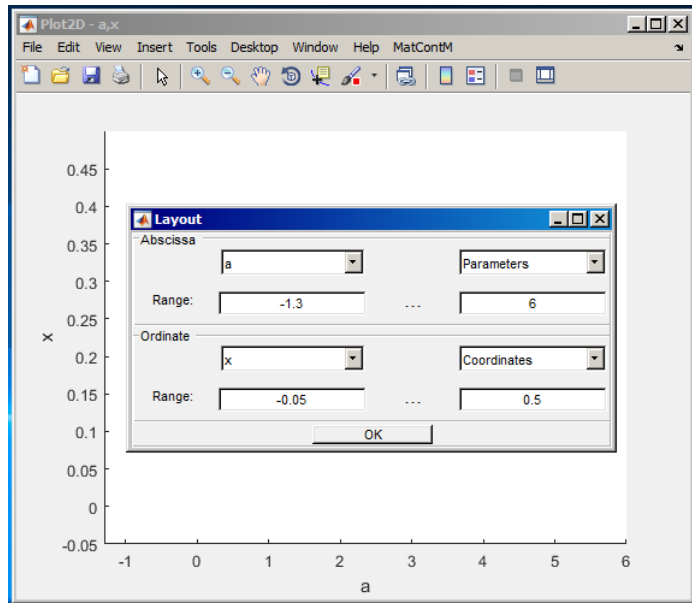


Figure 4

Select **Compute|Forward** in the main window and stop the computation after detecting the first **Branch Point (BP)**. This occurs at $a = 1$. A BP occurs generically when two fixed-point curves intersect. Select the BP point. The **Initial Point Type** in the main window is now set to **Branch Point**. The initializer `init_BPm.FPm` (selected by default) allows us to continue the intersecting curve of fixed points. We note that when a point is selected on a curve then the **Starter** and **Continuer** data of new curves started from that point are by default those of the curve on which the point was selected. Of course the user can change these.

Select **Compute|Forward** and keep an eye on the **Numeric** window. Stop the computation after detecting the first BP point (at $a = \frac{5}{3}$). The fixed points of this new branch are F_2 fixed points. Select the latest BP point and select **Compute|Forward**. This branch consists of F_3 fixed points. Select **Compute|Forward** if necessary but stop the computation after detecting a **NS** bifurcation at $a = 5$.

The resulting diagram is shown in Figure 5.

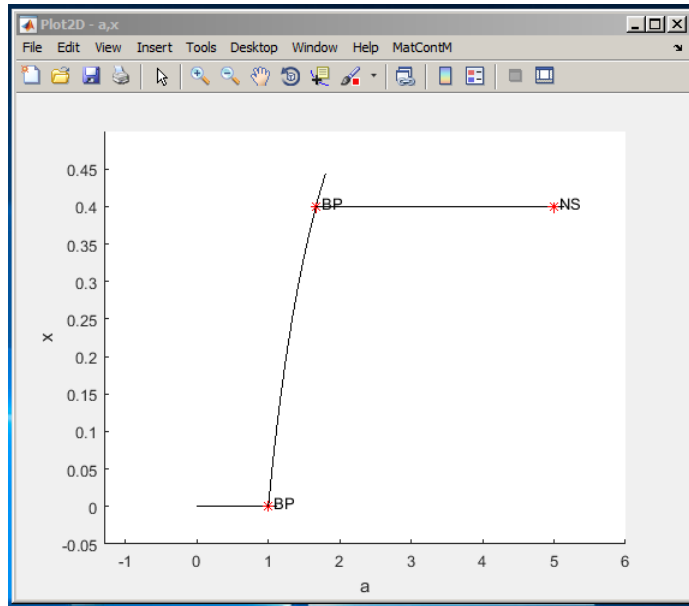


Figure 5

Section 5 demonstrates an alternative for obtaining this NS bifurcation.

3 Bifurcation curves

Select the **Neimark-Sacker** (NS) bifurcation point.

Before we start computing bifurcation curves, we will first make a change in our diagram settings. Curves are grouped together in what we call ‘diagrams’. The current diagram in which the curves are stored is the default diagram named `diagram`. We can create a new diagram using the context menu (right-click) in the **Current Curve** part of the main window, select **New Diagram** and enter the name `bifcurves`. All new curves will now be stored in that diagram.

The **Neimark-Sacker** bifurcation is now selected for continuation (see main window), a **Neimark-Sacker** bifurcation curve has been selected as curve (`init.NSm_NSm`). Note that the values in the new **Starter** and **Continuer** window are by default those of the fixed point curve on which the **Neimark-Sacker** point was detected.

Go to the **Starter** window and select a and ϵ as the two active parameters (Figure 6).

Create a new `plot2D` window (you may close the other plot window(s)). We will visualize the (ϵ, a) parameter plane. Go to **MatContM|Layout** in the plot menu of the `Plot2D` window and select parameter `epsilon` for the **Abscissa** and parameter `a` for the **Ordinate**. A comfortable starting range for the **abscissa** is $[-5, 4]$ and for the **ordinate** we select $[-2, 18]$ (see Figure 7).

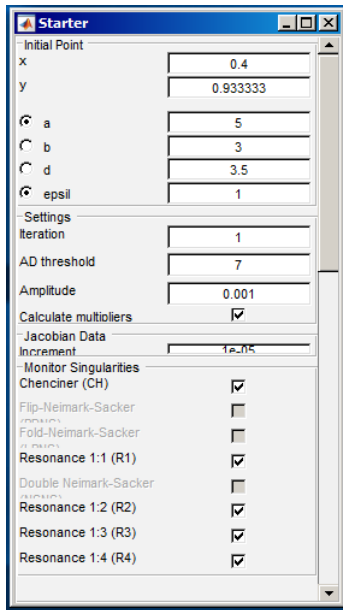


Figure 6

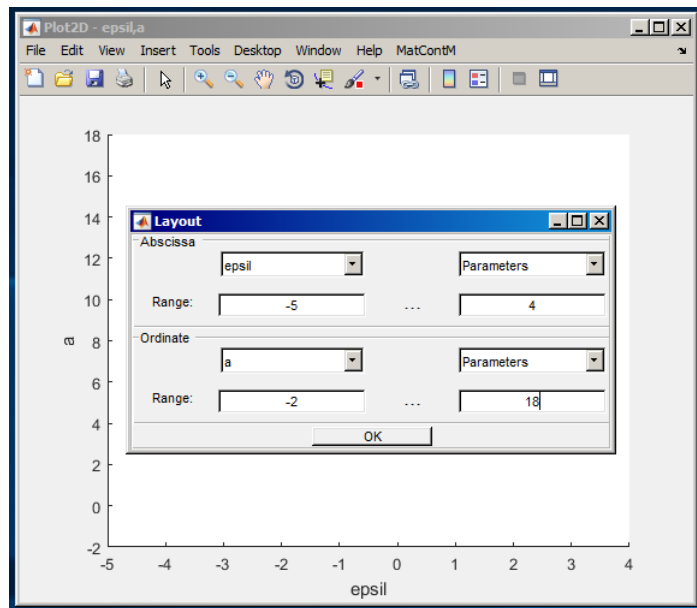


Figure 7

In the next NS continuation we will need to compute the codimension 2 bifurcation points to high accuracy. Therefore, change the tolerance variables in the Continuer window to $\text{VarTolerance}=1\text{e-}09$, $\text{FunTolerance}=1\text{e-}09$, $\text{TestTolerance}=1\text{e-}08$. Also, set $\text{InitStepsize}=0.001$ and $\text{MaxStepsize}=0.01$. This will increase accuracy, but decrease speed.

Start the continuation: Select **Compute|Forward** in the main window. You should detect several codimension 2 bifurcations. A **Resonance 1:4 (R4)** bifurcation, a **Resonance 1:3 (R3)** bifurcation and a **Resonance 1:2 (R2)** bifurcation (keep pressing **Resume** button). Select **Compute|Backward** and find a **Chenciner (CH)** bifurcation (see Figure 8).

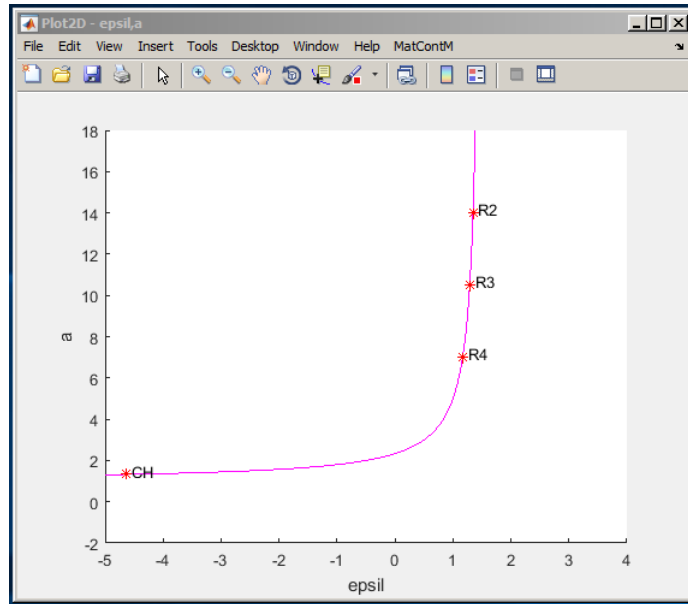


Figure 8

These codimension 2 bifurcations are *organizing centers*, allowing us to switch to other bifurcation curves.

Select the R2 bifurcation. You can do this by double-clicking on the graph or by using the **View Curve** browser. This codimension 2 bifurcation occurs when a Neimark-Sacker curve crosses a Period-doubling curve¹. We will compute that PD-curve. Selecting the R2 bifurcation allows us to select a number of branches seen in Figure 9.

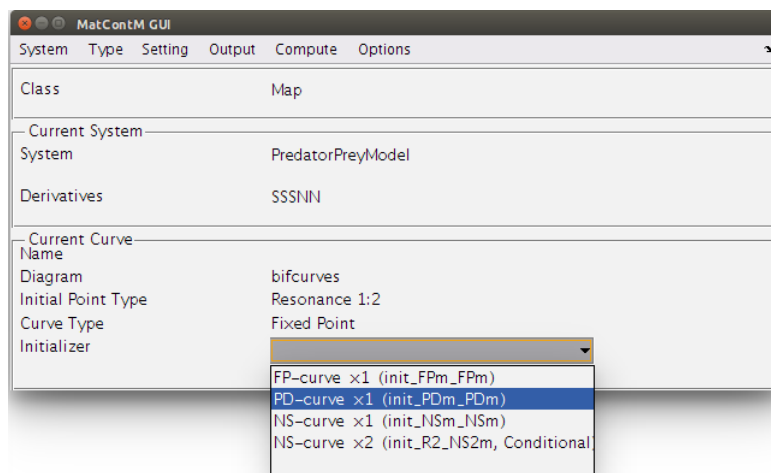


Figure 9

¹At this point, the Neimark-Sacker curve turns into the *neutral saddle* curve.

Select PD-Curve x_1 as initializer. The Starter data is inherited from the Neimark-Sacker curve, a and epsil should still be selected as the two free parameters. Restore the tolerance variables in the Continuer window to their default values $\text{VarTolerance}=1\text{e-}06$, $\text{FunTolerance}=1\text{e-}06$, $\text{TestTolerance}=1\text{e-}05$. Also, restore values $\text{InitStepsize}=0.01$ and $\text{MaxStepsize}=0.1$.

Select **Compute|Forward** and find a Fold-Flip (LPPD) bifurcation; resume computations. Select **Compute|Backward** and the R2 point gets rediscovered, after which you resume computations. The result is shown in Figure 10.

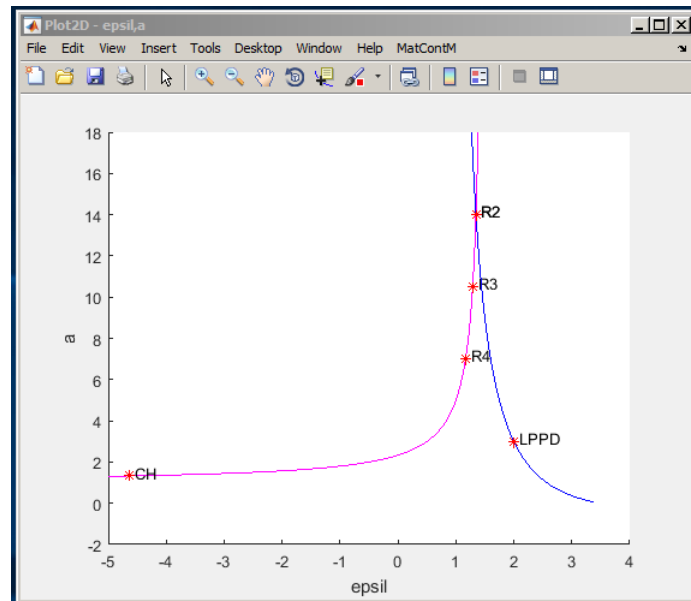


Figure 10

Select R4 on the Neimark-Sacker curve. There are multiple branches available for continuation. Under the right condition, two LP curves of the fourth iterate (LP^4) can be computed.

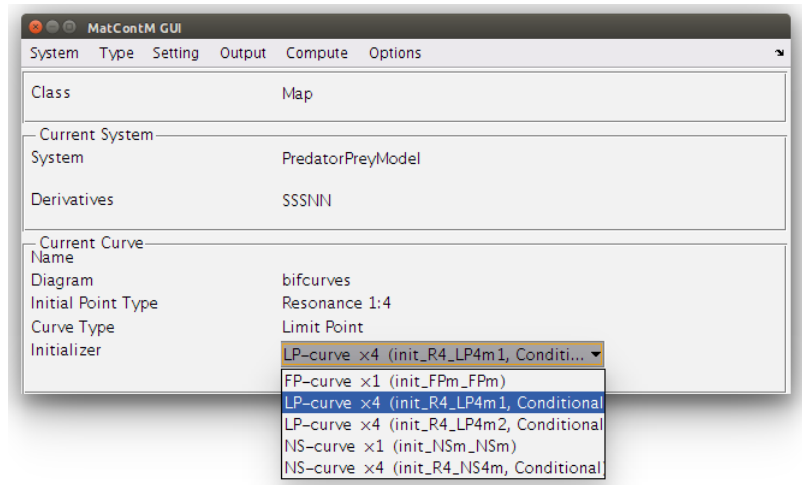


Figure 11

Select the `init_R4_LP4m1` initializer, which sets up a conditional branch. In this situation, the condition is met and this bifurcation curve exists. If conditions are not met you will be warned once you start the continuation.

Restore the tolerance variables in the **Continuer** window to their default values, i.e. `VarTolerance=1e-06`, `FunTolerance=1e-06`, `TestTolerance=1e-05`.

Restore `InitStepsize=0.01` and `MaxStepsize=0.1` and select **Compute|Forward**

You should find two LPPD bifurcations, a R1 bifurcation, one Cusp (CP) bifurcation and then the continuer will give up due to *'stepsize too small'*.

The result is shown in Figure 12; you can adjust the axis by using **MatContM|Layout** (or by using the MATLAB *pan tool* in the toolbar).

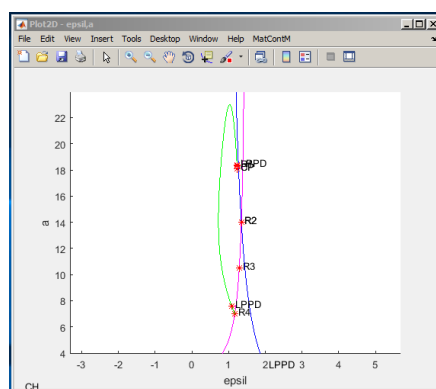


Figure 12: Computing the first LP4-curve from R4. The *pan tool* is selected in the toolbar.

Revisit the R4 bifurcation on the Neimark-Sacker curve. It should still be selected as the initial point in the main window. Choose `init_R4_LP4m2` as initializer to compute the

second LP^4 and select **Compute|Forward**.

You should find a LPPD point. After computation, you can extend this curve by selecting **Compute|Extend**. The continuation starts where it left off, the result is appended to the existing curve and is not stored in a new curve. We note that a change in any of the fields in the **Starter** or **Continuer** windows will be ignored in the computation of an extension.

If you extend several times you will find another LPPD bifurcation and a **Resonance 1:1 (R1)** bifurcation.

You can rescale your plot window with the MATLAB tools and the **Layout** window. When you change the scale of the graph, you might want to redraw all curves because the labels seem misplaced. By using the **MatContM** menu in the **Plot2D** window, you can select **Redraw Diagram** to redraw all the curves in the current diagram. **Redraw Curve** only redraws the curve selected in the main window.

You can use these two draw buttons to visualize curves after they have been computed and stored. You just need to load the desired curve as current curve, open and configure a plot window and press **Redraw Curve/Diagram**.

The result is shown in Figure 13. You see two green LP curves emanating from the R4 point on the NS curve.

We now select the first LPPD bifurcation, located on the first LP^4 -curve. The easiest way is to double-click on the LPPD label. Make sure all tools are disabled. This bifurcation lies on a LP curve of the 4th iterated map. This bifurcation indicates the presence of a PD curve passing through the LPPD point. Select the PD-curve x1 initializer and restore the tolerance variables in the **Continuer** window to their default values **VarTolerance=1e-06**, **FunTolerance=1e-06**, **TestTolerance=1e-05**. Restore **InitStepsize=0.01** and **MaxStepsize=0.1**. Perform a forward continuation, followed by a backwards continuation.

The loop emanating from R3 is computed by double-clicking on R3 and computing NS-curve x3 after restoring the tolerance variables and restoring **InitStepsize=0.01** and **MaxStepsize=0.1**.

The full picture of the **bifcurves** diagram is shown in Figure 14. The blue curve in this Figure is the PD^4 curve. The closed magenta curve is the NS^3 curve. Figure 15 is a zoom near the R3 and R4 points; Figure 16 is a more detailed zoom near the R4 point.

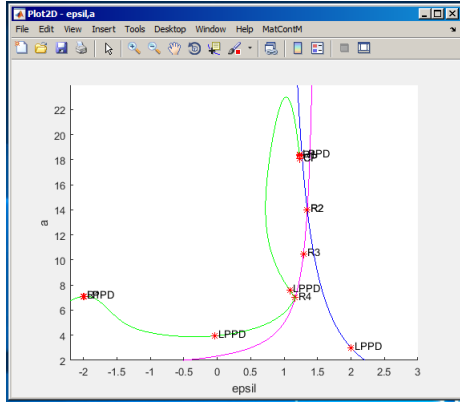


Figure 13

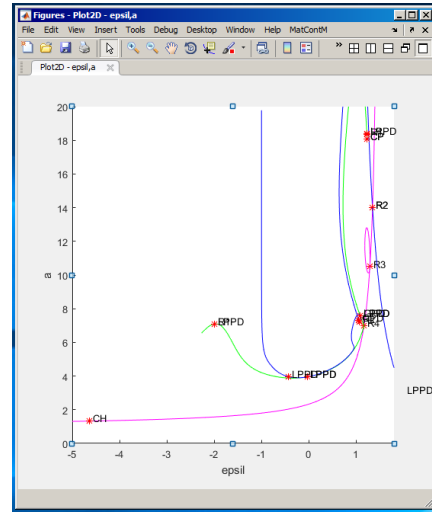


Figure 14: All the bifurcation curves in the diagram bifcurves

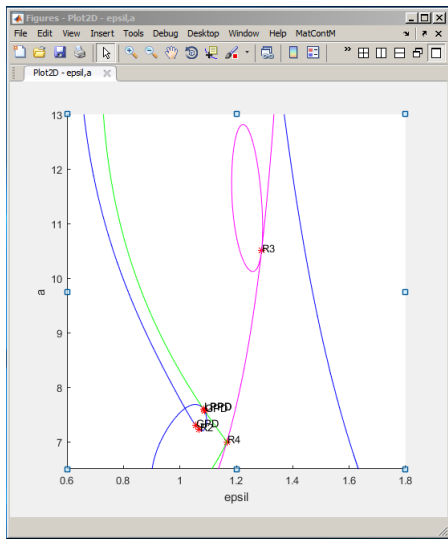


Figure 15: Zoom containing R3 and R4

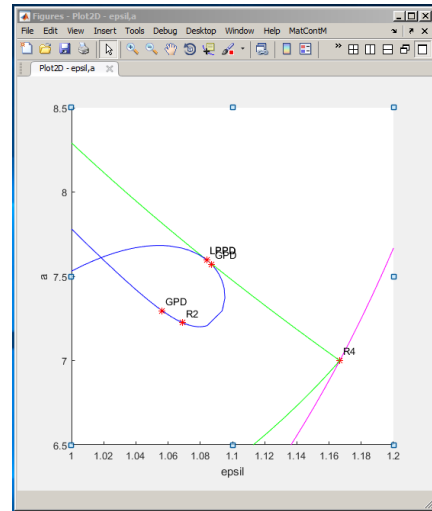


Figure 16: Zoom near R4

To create a three-dimensional plot, select **Output|Graphic|3D Plot**. Further select **MatContM|Layout** in the Plot3D window and use the layout settings as in Figure 17. Select **MatContM|Redraw Diagram** to draw all computed bifurcation curves. These are the curves located in the diagram bifcurves.

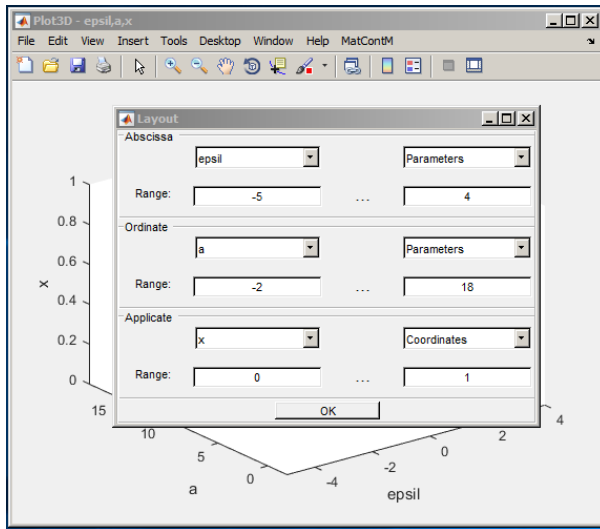


Figure 17

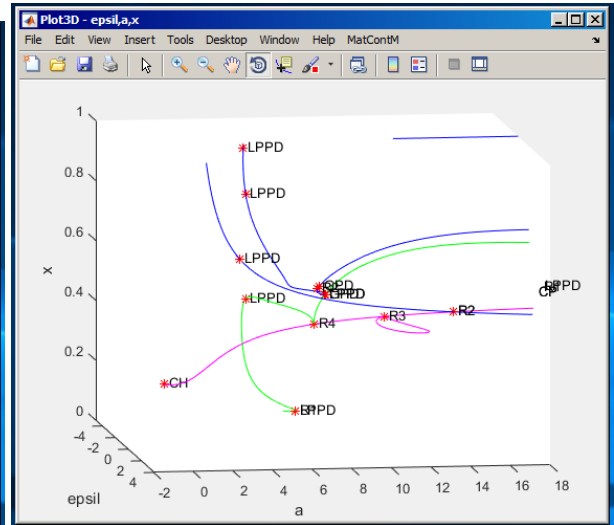


Figure 18: The *rotation tool* has been selected.

4 User functions

User functions are custom test functions added by the user to a system. The continuation algorithm evaluates these functions at each point on the computed curve and checks for sign changes in order to detect zeros. When a zero is detected, a bisection-like algorithm is used to determine the exact point where the zero occurs. These zero points are reported as special points. A user function can use the coordinates and parameters and returns a real value. Such functions can be used to detect situations specific to the system or to add flags for certain values.

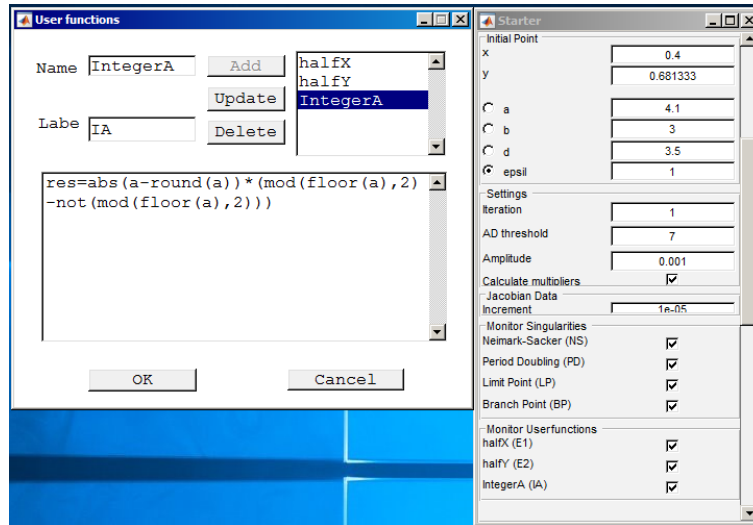


Figure 19: The User functions window and the Starter window after entering user functions.

We can add, edit and delete user functions by going to the **System Browser** by selecting **System|Systembrowser**. Select the system **PPModel** and click on the button **Add/Edit Userfunctions** below. The window for user functions appears (Figure 19). We can now add, edit and delete user functions for the selected system. We note that no check is done on the input. So if the input is not grammatically correct in MATLAB then no error or warning will be generated but the session will probably crash later on.

For example, we add 3 user functions. We wish to know when a coordinate (population density) reaches the value 0.5. We start by filling in the **Name** field, we will name our function **halfX**. We fill in the label by entering **E1**. The formula for our user function has to be entered in the form: **res=...** (no spaces after **res**). We can use the coordinates x and y and the parameters a , b , d and ϵ in our formulas.

We enter **res= x - 0.5**.

Click on the **Add** button to add the user function to the list in the right upper corner.

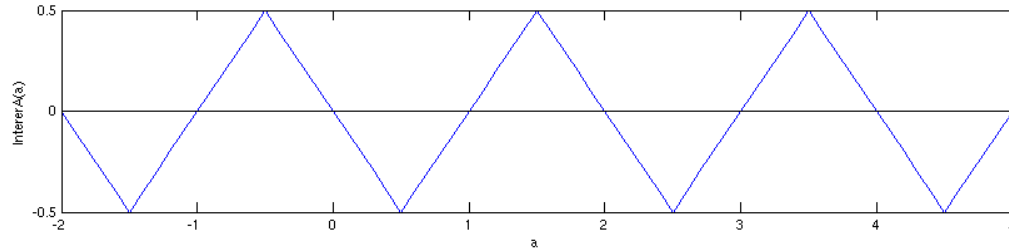
We name the second function **halfY**, the label is **E2** and the equation is **res=y - 0.5**. Press **Add**.

Our third function allows us to monitor when the parameter a has an integer value during computation.

We must design a function where $f(a) = 0$ if a is an integer. A sign change must occur in the zero point for it to be detected by the continuation algorithm. We can use the MATLAB built-in functions. We name the function **IntegerA**, give it label **IA** and use the equation:

$$\text{res}=\text{abs}(a - \text{round}(a))*(\text{mod}(\text{floor}(a),2) - \text{not}(\text{mod}(\text{floor}(a),2)))$$

The function values for the interval $[-2, 5]$ are shown in the following plot:



Press **Add**. Press **OK** to exit the window, the user functions are now specified.

You can always return to add, edit and delete user functions. You can edit a function by selecting it on the list, make changes, and press **Update**. Make sure the system is loaded as the **Current System** in the main window.

Now go to the **Starter** window (Figure 19). The user functions are now added to the sub panel named **Monitor Userfunctions**, you can enable them or disable them. They are disabled by default so enable them all.

Enabled user functions become available in the **Plot Layout** window and in the **Numeric** window (see Figure 20).

Using the settings from Figure 19 we compute a fixed-point curve by varying the parameter ϵ .

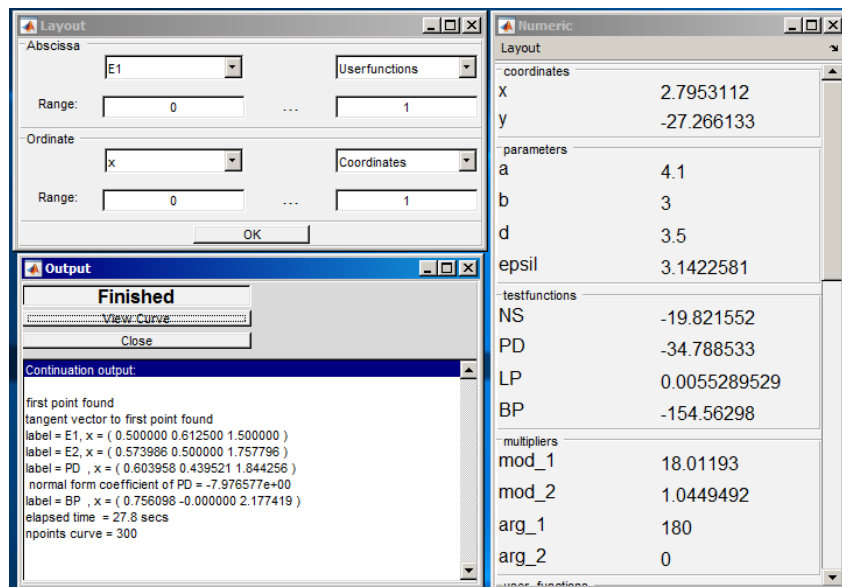


Figure 20: Numeric and Layout window with user functions. The continuer output displays the detection of user function zeros.

User function zeros are treated in the same way as the bifurcations, they are displayed on plots and they can be selected in the **Data Window** when viewing a curve. When a user function zero is selected as initial point for continuation, no initializers (branches) are available. This can be solved by declaring the type of the point, by selecting in the menu: **Type|Initial Point|**. The results of the computation starting from the setting in Figure 19 are shown in Figure 21. Another computation is shown and Figure 22 and shows what happens if two bifurcation points coincide.

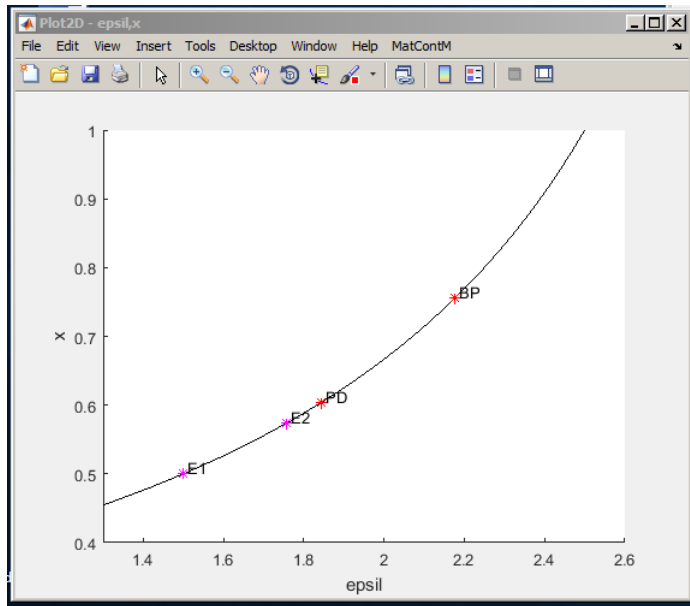


Figure 21

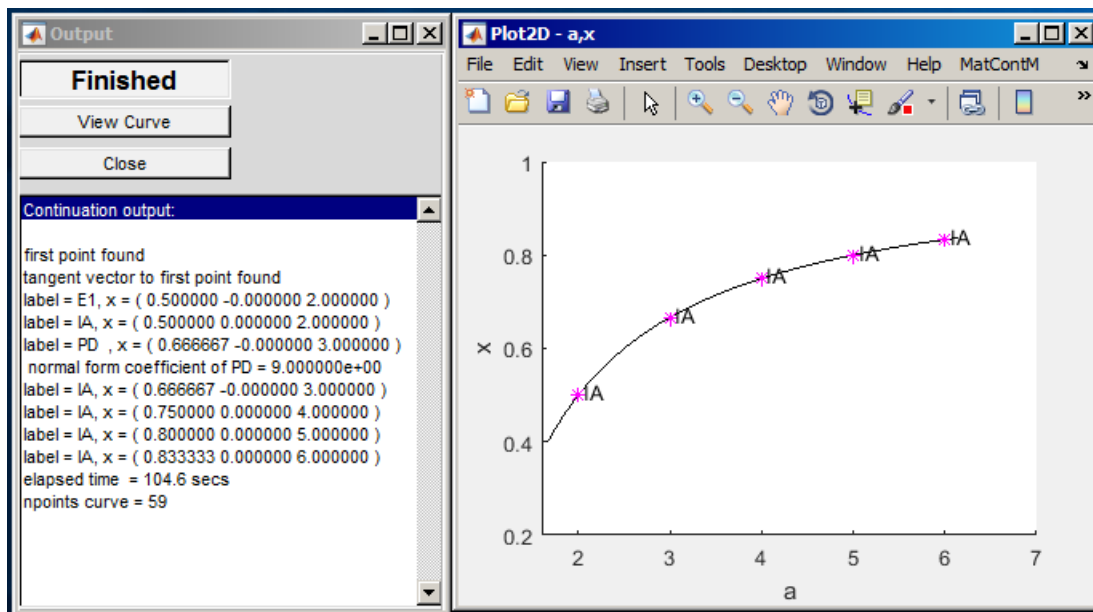


Figure 22

5 Configuring the Starter window from the MATLAB Command Window

Working with a graphical user interface has many advantages but also some disadvantages. One of the difficulties is entering many numbers in the **Starter** window. Once the continuation has started and bifurcations are being detected, typing becomes minimal.

While the GUI of MatContM is running, the MATLAB command line stays available. The `CommandLineInterface` bridges the gap between GUI and command line, it allows for the transfer of data between the GUI and the command line workspace. We will show this by example for system (1).

First, select in the main window menu: **Type|Initial Point|Fixed Point**, to prepare for a fixed point continuation. The map `PPModel` (`PredatorPreyModel`) has three types of fixed points. F_1 and F_2 are simple to work with. F_3 is rather complicated. We will use the MATLAB *Anonymous Functions* features to make computing the coordinates easy for different sets of parameters.

```
>> F3_point = @(a,b,d,epsil) [1/(d-epsil) , (d/(d-epsil))*((a/b)*(1-(1/(d-epsil))))-(1/b)];
>> F3 = @(Param) F3_point(Param(1) , Param(2) , Param(3) , Param(4));
```

Note that you can copy-paste this if you are using a pdf viewer. We can now use the `F3_point`-function to compute the coordinates for any parameters or use the `F3`-function when the parameters are stored in an array.

```
>> param = [4.1, 3, 3.5, 1];
```

```
>> coord = F3(param)
```

```
coord =
```

```
    0.4000    0.6813
```

Make sure system (1) is loaded and the initial point type is set to **Fixed Point** and the initializer is set to **FP-Curve x1**. **Iteration** in the **Starter** window should be set to 1.

While the GUI is running, we execute in the MATLAB prompt:

```
>> cli = CommandLineInterface;
```

This creates a `CommandLineInterface` object that is stored in `cli`. This object is tethered to the current GUI session. If the GUI session is closed, `cli` becomes invalid. If the GUI is restarted a new `cli` object has to be created. You can use `cli` throughout the current session of the GUI.

We will transfer the data to the GUI:

```
>> cli.setPa(param);           %or: cli.setParameters(param);
>> cli.setCo(coord);          %or: cli.setCoordinates(coord);
>> %cli.setIP(coord, param) %alternative command, IP = InitialPoint
```

The parameters and coordinates in the **Starter** window should now be updated. Use `cli.getCurve()` to retrieve the raw continuation data after a computation.

If you select a as the free parameter and do a forward computation, you should find a Neimark-Sacker bifurcation at $a = 5$.

6 Additional problems

- A. **Bifurcation curves of delayed logistic map** Revisit the discrete-time dynamical system generated by the map

$$\text{DelLogMap} : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} rx(1-y) + \epsilon \\ x \end{pmatrix}$$

studied in Tutorial MI. Use the located bifurcation points on the stored curves as initial data for the two-parameter continuation of these bifurcations to obtain Figure 23.

- B. **Discrete-time Lotka-Volterra model**

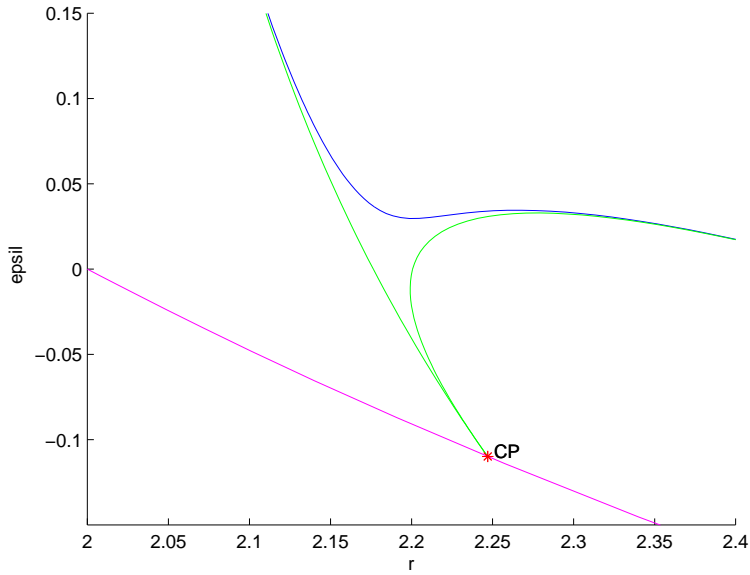


Figure 23: Bifurcation curves of DLM: NS (magenta), LPC for the period-7 cycles (green), and PD for the period-7 cycles (blue). The point labeled as CP is actually close to the resonance 1:7 point on the Neimark-Sacker curve NS.

Consider the following simple prey-predator population model²

$$\begin{cases} x_{k+1} = ax_k(1 - x_k) - bx_ky_k, \\ y_{k+1} = dx_ky_k \end{cases}$$

where (x_k, y_k) are the prey and predator densities in year k , and (a, b, d) are positive parameters. Study fixed points and cycles of the corresponding planar map

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} ax(1 - x) - bxy \\ dxy \end{pmatrix}.$$

1. Study the map for $b = 0.2$ and $(a, d) \in [0, 12] \times [0, 4]$ using MatContM:
 - (i) Find by simulations a positive fixed point of the map at $a = 3.5$ and $d = 2.67$. Continue this fixed point w.r.t. parameter d and detect two codim 1 bifurcations of this point, namely: NS and PD.
 - (ii) Continue the corresponding codim 1 bifurcation curves $NS^{(1)}$ and $PD^{(1)}$ in the (d, a) -plane and locate several codim 2 points, including strong resonances R2, R3, and R4 on the $NS^{(1)}$ curve.
 - (iii) Starting from the detected R4 point, compute two $LPC^{(4)}$ bifurcation curves of 4-cycles rooted there.

²The map (1) reduces to this model for $\epsilon = 0$.

- (iv) Compute the period-doubling $PD^{(4)}$ and Neimark-Sacker $NS^{(4)}$ bifurcation curves of 4-cycles located between the $LPC^{(4)}$ curves.

Hint: To obtain starting points, first continue a stable period-4 cycle from one of the LPC points w.r.t. parameter a and stop the computation somewhere between the LPC-curves. Then continue the last stable 4-cycles w.r.t. parameter d to detect its **NS** and **PD** bifurcations.

2. Study the map analytically for positive (a, b, d) :

- (i) Find all non-negative fixed points and derive explicit formulas for the $NS^{(1)}$ and $PD^{(1)}$ curves, as well as bifurcation curves of fixed points with at least one zero coordinate. What is the stability domain in the (d, a) -plane of the positive fixed point ?
- (ii) Compute analytically the normal form coefficients for the PD and NS bifurcations along the curves $NS^{(1)}$ and $PD^{(1)}$. Predict what happens when crossing these curves at generic points.

Are your numerical and analytical results in agreement with each other ?