New MatCont and a numerical bifurcation study of a perception problem in psychophysics

Willy Govaerts*, Yuri A. Kuznetsov†‡, Hil G.E. Meijer ‡, Niels Neirynck* and Richard J.A. van Wezel††

*Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Belgium

†Department of Mathematics, Utrecht University, The Netherlands

‡Department of Applied Mathematics, Twente University, The Netherlands

†† Donders Institute for Brain, Cognition and Behaviour, Radboud University, Nijmegen, The Netherlands

Summary. We discuss the new GUI environment of the MATLAB software package MatCont for numerical bifurcation studies of continuous dynamical systems. It is built upon the corresponding command line package Cl_MatCont. The package is freely available via sourceforge.net/projects/matcont and offers both interfaces. Mathematically, the functionalities of MatCont with respect to bifurcation techniques are unrivalled. For instance, no other software allows to compute the normal forms of codimension two bifurcations of periodic orbits, or to start curves of codimension one bifurcations of periodic orbits from codimension two equilibrium points. Though widely used, the previous version of MatCont was at the end of its life span of maintainability and the new MatCont gives it a fresh start. It is completely reorganized with a better documentation and an improved data handling with a Data Browser, a Diagram Organizer and a Spreadsheet Viewer. Other new features are the Command Line Interface, the functionality of computing Poincaré maps and many facilities to simplify the use of the software. As an application we discuss a computational model that describes the stabilization of percept choices under intermittent viewing of an ambiguous visual stimulus at long interstimulus intervals. Unlike previous studies we incorporate the time that the stimulus is on (T_{on}) and off (T_{off}) explicitly as bifurcation parameters of the model. We compute the bifurcations of periodic orbits responsible for switching between alternating and repetitive sequences. We show that the region of bistability of repeating and alternating behavior is a wedge in the parameter plane bounded by two curves of limit point bifurcations of periodic orbits and one curve of period-doubling bifurcations.

Introduction

We consider smooth continuous dynamical systems of the form

$$\frac{dx}{dt} \equiv \dot{x} = f(x, \alpha), \quad x \in \mathbb{R}^n,\ \alpha \in \mathbb{R}^m, \tag{1}$$

with state variable x, parameter α and f a sufficiently smooth function (continuous derivatives up to order 5 are needed in some cases). The numerical bifurcation analysis of (1) requires a dedicated software package. For this purpose MATCONT was developed, a MATLAB continuation toolbox available at *http://sourceforge.net/projects/matcont/*. It is a successor package to CONTENT [7] and LINBLF [6].

Bifurcation analysis usually starts with equilibria and periodic orbits (limit cycles). The stable ones can be found by time integration of the system, see the MATCONT manual [4], §6.2 and §7.4. By numerical continuation under variation of a single system parameter one can detect and study the codimension one bifurcations, i.e. limit points and Hopf points for equilibria, limit points of cycles, period-doubling and Neimark-Sacker (torus) bifurcation points for periodic orbits. Further continuation of these codimension one bifurcations under variation of two system parameters leads to the detection and study of codimension two bifurcation points; there are 5 codimension two types of bifurcations of equilibria and 11 types for periodic orbits. MATCONT allows to study all these bifurcations numerically and perform many related tasks, including the study of orbits homoclinic to saddle, homoclinic to saddle-node and heteroclinic orbits. Critical normal form coefficients are computed at bifurcation points. For this we rely on symbolic derivatives or if these are not available, on finite difference approximations. Bifurcation curves are defined by a system of equations consisting of fixed point and bifurcation conditions. The continuation curves can be visualized using the plot capabilities of the GUI; this can be done during and after the continuation. Special windows are provided to help with maintaining systems, diagrams and curves when generating a large amount of data.

Earlier versions of MATCONT and their functionalities were described in [1] and [2]. We will restrict to the new features in the renovated 2019 environment MATCONT7.1 and later versions. From a computer science point of view MATCONT7.1 is a completely new creation. It has a clear separation of computational and control routines to facilitate the maintainability. Its GUI allows a maximal flexibility to reprogram the layout of windows, buttons and input fields on the screen. It contains automatic tests to check if a new MATLAB version produces the same results as the previous version. Error handling of plots is much improved so that malfunctioning of plots (for any reason) does not crash the computations. Unlike the previous versions, it has an external documentation ([8], Ch. 6) and a detailed internal documentation. The internal documentation of a source file is accessed by typing '`doc filename.m`' on the command line; a reference page is then generated based on the comments in the source file.

The core mechanism of the new MATCONT environment is an intermediate layer of routines between CL_MATCONT and the GUI as seen by the user. It is described in [8], Chapter 6; the main elements in this mechanism are the MATLAB

classes *settings, session* and *solution*, cf. the section **Command Line Interface.** This layer also protects against nonsense input in the CL_MATCONT routines.
Important other parts, which can be used semi-autonomously are the generator of the system m-files (*SysGUI.m*), the spreadsheet viewer (*GUISimCurveTable.m* and *GUIContCurveTable.m*) and the GUI subsystems *Data Browser* and *Diagram Organizer*, which are stored in subfolders of the *GUI* folder. The main driver files are `matcont.m`, `GUI/MATCONTGUI.m` and `GUI/Session.m`.
For the practical use of MATCONT it is best to start with the tutorials which are provided with the software; the manual [4] is a good reference to the command line version CL_MATCONT.
To study a system in MATCONT one has to describe it in a *system m-file,* which serves as a handle to the system. MATCONT provides an interface to build such m-files, see the manual [4], §4 or the first tutorial.
The MATCONT panels are described in [8], Ch. 5. The main MATCONT panel is shown in Figure 1. We note in particular the tab line at the top with the six tabs **Select, Type, Window/Output, Compute, Options,** and **Help.**

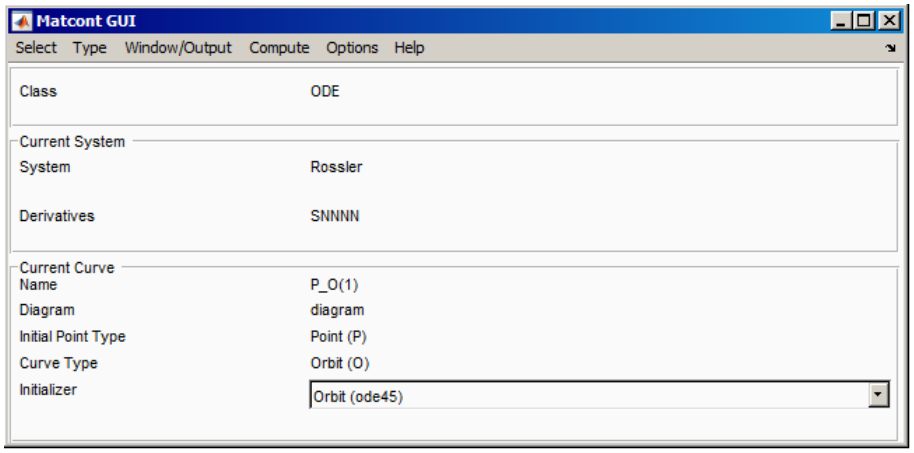

Figure 1: The main MATCONT panel.

Data management in MATCONT

The Diagram Organizer

The database of MATCONT consists of an archive of systems one of which is the *current system*.
A system is internally characterized by a system m-file, a system mat-file and a system directory, all with the name of the system. They are all in the subdirectory **Systems** of MATCONT. When a MATCONT session is closed **(Select|Exit)** then the session information is stored in a file `session.mat` in the **Systems** directory. This allows to restart the MATLAB session at the point where it was stopped.
The m-file is a readable file that contains all defining information on the system; the mat-file contains the same information in a structured way that is accessible to the MATCONT software. Both stay unchanged as long as the system is not changed or deleted.
The directory of each system also contains a file `session.mat` which contains the information that is necessary to restart or reproduce the computations on that system at the stage where it was left, including the position and contents of all windows. However, plots have to be redrawn.
The system directory also has at least one default subdirectory called `diagram`. This and other subdirectories of the system directory are called *diagrams.* Each diagram contains a number of mat-files and each mat-file describes a computed curve with enough information to recompute the curve. Each computed curve contains a number of special points, which includes the first point, the last point, bifurcation points, and zeros of userfunctions but other entities may also be defined as special points. An important example of this is the case of an orbit where a **Select Cycle** object is identified as a special point (for more details see the section **Other New Features**).
Clicking **Select|Organize Diagrams** in the main MATCONT panel opens the Diagram Organizer which allows to move curves from one diagram to another, see Figure 2.

The Spreadsheet Viewer

The Spreadsheet Viewer allows to inspect all stored data of a computed curve. It can be accessed by pressing the **View Curve** button in a Curve window that is opened in the Data Browser. However, it is not possible to load data from the Spreadsheet Viewer for further use in MATCONT.

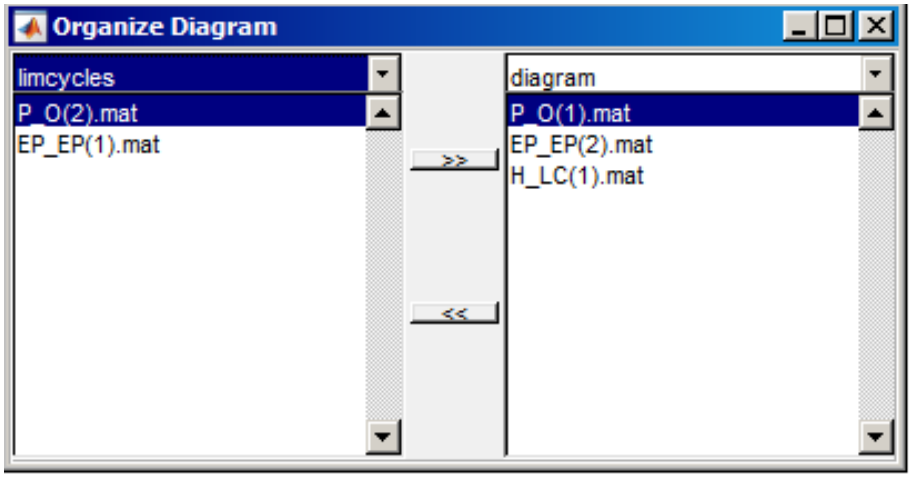

Figure 2: The Diagram Organizer panel.

The Command Line Interface

The Command Line Interface (cli) is a collection of MATLAB commands that allow to use practically all GUI functionalities from the command line in an interactive way.
This has a number of advantages. First, if the number of state variables or parameters is large and provided by an external file, then it is possible to load them as vectors directly into MATCONT. Second, it allows to write a MATLAB script that uses MATCONT in a semi-automatic way, for example to perform a bifurcation analysis for a range of values of parameters which are fixed in each particular bifurcation analysis. Third, it allows to load the raw computed data of a continuation or simulation directly into the Matlab workspace.
In the cli we address directly the intermediate layer of routines. This requires to set one or more of the class variables `settings, session, solution` as global.
To illustrate the use of the cli we introduce the *Rössler* system [10]:

$$\begin{cases} \dot{x} &= -y - z \\ \dot{y} &= x + Ay \\ \dot{z} &= Bx - Cz + xz, \end{cases} \tag{2}$$

where (x, y, z) are the phase variables, and (A, B, C) are the parameters. We introduce it in MATCONT under the name *ROESSLERTest* and let the derivatives of order 1, 2 and 3 be computed symbolically. To perform a time integration we select the initial point type 'Point' and input $x = 1$, $y = 2$, $z = 3$, $A = 0.2$, $B = 0.4$, $C = 0.7$ as initial values. In the **Integrator** window we set the interval to 200. We then turn to the MATLAB command line to inspect the settings:

```
>> global settings
>> settings
settings =
              system: ROESSLERTest
                  IP: Point (P)

        option_pause: At Special Points
      option_archive: 2
       option_output: 1
============================
                time: 0
                co_x: 1
                co_y: 2
                co_z: 3
               coord: [ 1, 2, 3 ]

          parameters: [ 0.2, 0.4, 0.7 ]
                pa_A: 0.2
                pa_B: 0.4
                pa_C: 0.7
============================
            Interval: 200
```

```
       eventfunction: <disabled>
    InitStepSize_sim: <automatic>
     MaxStepSize_sim: <automatic>
        RelTolerance: 0.001
        AbsTolerance: 1e-06
              Refine: 4
         Normcontrol: false

>> settings.co_x
ans =
     1
>> settings.coord
ans =
     1     2     3
```

We can also change some of the settings:

```
>> settings.coord.set([-5 5 10])
>> settings.parameters.set([0 0.4 4.5])
```

In the GUI we can check that these changes are visible in the **Starter** window. There we can also perform the computation. The outcome is stored in the `solution` class. We inspect it in the MATLAB command window:

```
>> global solution
>> solution
solution=
   SimCompSolution with properties:
            t: [1477x1 double]
            y: [1477x3 double]
       method: @ode45
        tspan: [0 200]
      options: [1x1 struct]
        param: [0 0.4000 4.5000]
           tE: []
           yE: []
           iE: []

>> solution.y
ans=
-5.0000 5.0000 10.0000
-5.0763 4.9739  9.5078
...
```

We return to the GUI, select the last point of the computed orbit (use the **View Result** button in the **Control** panel) and declare it to be an equilibrium. By default the Curve Type will now be 'Equilibrium'. We again inspect the `settings`:

```
>> settings
settings =
              system: ROESSLERTest
                  IP: Equilibrium (EP)

============================
        InitStepsize: 0.01
         MinStepsize: 1e-05
         MaxStepsize: 0.1

      MaxNewtonIters: 3
        MaxCorrIters: 10
        MaxTestIters: 10
        VarTolerance: 1e-06
        FunTolerance: 1e-06
       TestTolerance: 1e-05
```

```
               Adapt: 3

        MaxNumPoints: 300
         CheckClosed: 50
============================
               co_x: -0.000335576810360529
               co_y: -0.000180608885605843
               co_z: -3.20127988121691e-05
              coord: [ -0.00033557681, -0.00018060888, -3.2012798e-05 ]

          parameters: [ 0, 0.4, 4.5 ]
                pa_A: 0
                pa_B: 0.4
                pa_C: 4.5
         pa_A_select: false
         pa_B_select: false
         pa_C_select: false

          test_EP_BP: true
           test_EP_H: true
          test_EP_LP: true

         eigenvalues: true
============================
```

We need to select an active parameter in the equilibrium continuation. This can be done via the **Starter** window or via the command line:

```
>> settings.pa_A_select.set(true)
```

We will also restrict the number of computed points to 49. This can be done either via the **Continuer** window or via the command line:

```
>> settings.MaxNumPoints.set(49)
```

We execute the continuation in the GUI. The solution data can be obtained in the MATLAB Command window:

```
>> solution
solution=
ContCurve with properties:

x: [4x49 double]
v: [4x49 double]
s: [3x1 struct]
h: [5x49 double]
f: [3x49 double]

>> size(solution.x)
ans=
     4      49
```

So far all driving steps were executed in the GUI, where they were relegated to the `session` class. In a more advanced use of the cli we can also perform these steps from the command line by declaring the `session` class global. We then add the commands:

```
>> global session
>> session
```

The `session` output recalls the settings and then offers a choice of three buttons, labeled

```
View Computations  View Actions  View Switches
```

Clicking the first button is equivalent to executing the command

```
>> session.select()
```

It provides the numbered list of computations (usually time integrations or continuation runs) that can be selected from the given Initial Point type. They can be selected by clicking, or by typing

```
>> session.select(k)
```

where k is the list number.
Clicking the second button is equivalent to executing the command

```
>> session.compute()
```

It provides a fixed list of options, namely **Forward, Backward,** and **Extend**. The list can be handled as in the previous case.
Clicking the third button is equivalent to executing the command

```
>> session.switches()
```

It provides the list of objects that can be chosen as new Initial Points. The list can be handled as in the previous cases. One can change the type of an initial point from the command-line by using the instruction

```
>> session.changeInitPoint('H')  %force IP type to be 'Hopf (H)'
```

We note that it is always possible to get information about the current system:

```
>> settings.system
ans =
     CLSystem with properties:
name: 'RoesslerTest'
coordinates: {'x'  'y'  'z'}
  parameters: {'A'  'B'  'C'}
dim: 3
time: 't'
handle: @ROESSLERTest
userfunctions: {}
ufdata: []
diagramlocation: 'H:\MatCont7p1\Systems\ROESSLERTest'
derstr: 'SSSNN'
equations: [3x14 char]
```

Event functions and Poincaré maps

A Poincaré section is a (in general, curved) surface in phase space that cuts across the flow of a dynamical system. The Poincaré map transforms the Poincaré section onto itself by relating two consecutive intersection points. Only those intersection points count, which come from the same side of the section. In this way, a Poincaré map turns a continuous-time dynamical system into a discrete-time one. If the Poincaré section is carefully chosen no information is lost concerning the qualitative behaviour of the dynamics. For example, if the system's state is attracted to a limit cycle, one observes dots converging to a fixed point in the Poincaré section as in Figure 5.
When computing an orbit $(t, y(t))$ in MATLAB an event can be defined as going through a zero of a scalar event function $G(t, y)$. If G does not explicitly depend on time in an autonomous dynamical system, this functionality can be used to see the Poincaré map in action. However, one shortcoming of the MATLAB solvers is that they do not interactively report on these events.
In the 6.11 and earlier versions of MATCONT, Poincaré maps were therefore computed by using a specific curve type, called Discrete Orbit (DO). For this type of orbits the ODE solvers were manipulated to use an explicitly implemented approximation strategy to locate the Poincaré intersection points.
In the new GUI, there is no longer a specific curve type. In order to allow for interactive plotting of events, the plot routine monitors sign changes of the values of the event function. Whenever a sign change is observed, the solver is called again on a smaller part of the curve to extract the event. Due to the implementation of this workaround, the list of detected events after computation in rare cases might contain more events than were displayed on an interactive plot.
For the sake of generality MATLAB allows to define vector-valued event functions whereby each component function defines its own event. This is done by setting the Events property to a handle to a function, e.g. `@events` with the syntax
`[value,isterminal,direction] = events(t,y,varargin)`
where y is a state vector. The input variable `varargin` is a cell array that contains the values of the parameters. If parameters are explicitly used in the definition of the event function, then `varargin` should be replaced by an explicit list of parameter names.
If there are k event functions then for $i \in \{1, \ldots, k\}$:

- `value(i)` is the value of the $i-$th event function.
- `isterminal(i)` = 1 if the integration is to terminate at a zero of the $i-$th event function and 0 otherwise.
- `direction(i)` = 0 if all zeros of the $i-$th component are to be computed (the default), +1 if only the zeros are needed where the event function increases, and -1 if only the zeros are needed where the event function decreases.

The use of event functions to compute Poincaré maps in CL_MATCONT is discussed in the manual [4]. We now illustrate the GUI implementation by example. We consider again the *Rössler* system (2).
Suppose that we want to compute an orbit and detect two events along it, namely $x = 0.2$ and $y = 0.3$. We need only the events where x, respectively y, are increasing and the integration will not be terminated if an event is detected.
We then define an event function `testEV` as follows:

```
function [value,isterminal,direction]= testEV(t,y,varargin)
value=[y(1)-0.2;y(2)-0.3];
isterminal=zeros(2,1);
direction=ones(2,1);
end
```

The function `testEV.m` is placed in the MATCONT main directory (or anywhere else on the MATCONT path)
We integrate the *Rössler* system from 0 to 100 starting from the point $[-5; 5; 10]$ with parameter values $(0.25, 0.4, 4.5)$ and input the name 'testEV' (without quotes) in the EventFunction field of the **Integrator,** see Figure 3.

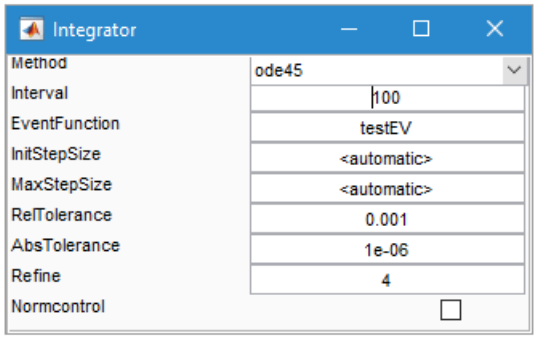

Figure 3: Integrator with an Event function.

We also open a MATCONT **Plot3D** window to display the state variables in the range $\{-9 \leq x \leq 9, -9 \leq y \leq 9, -2 \leq z \leq 11\}$. The 3D output is shown in Figure 4. We note that zeros of the first event function are marked as 'E1', those of the second as 'E2' and so on.
In Figure 5 we show the convergence of the iterates if the Poincaré map to a fixed point in a **Plot2D** window with $\{4.22 \leq x \leq 4.32, 2.3 \leq z \leq 2.8\}$,
The computed output is stored in the mat-file of the computed curve, from where it can be recovered for further use. It can also be sent directly to the MATLAB workspace by pressing the **Export** button in the **Data Browser** window that is opened by pressing the **View Curve** button in the **Control** window after the integration run. In the latter case we get the following output in the MATLAB command window :

```
exported=
  SimCompSolution with properties
         t: [1461x1 double]
 y: [1461x3 double]
  method: @ode45
 tspan: [0 100]
   options: [1x1 struct]
  param: [0.2500 0.4000 4.5000]
   tE: [33x1 double]
 yE: [33x3 double]
 iE: [33x1 double]
```

Here `tE` is the vector of time points where events were discovered, `yE` is the corresponding matrix of state variables and `iE` is a vector of ones and twos which refer to either the first or the second event.
Event points can also be selected as special points in the data browser and by double-clicking on the labels in the plots.

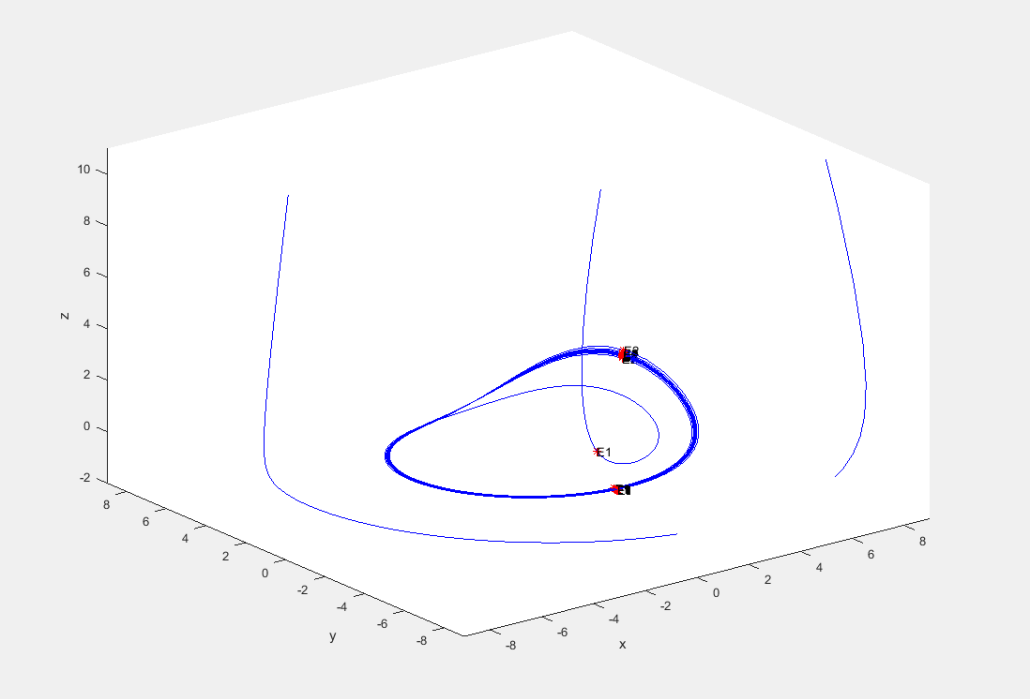

Figure 4: Time integration with event points that converge to fixed points.

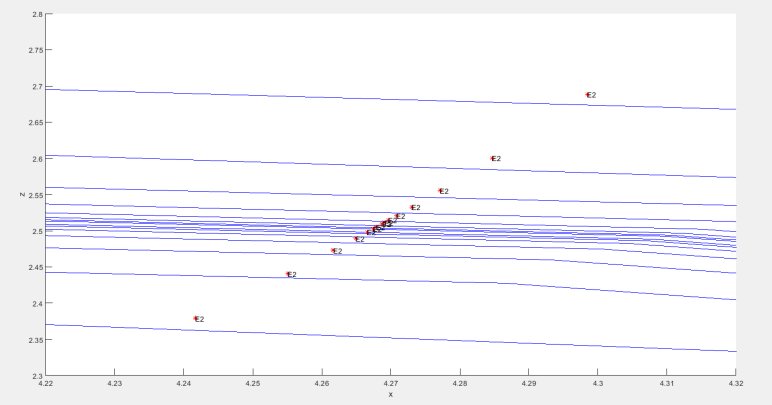

Figure 5: Convergence of the iterates of the Poincaré mapto a fixed point, alternating between points at the bottom left side and the top right side.

Other new features

The new MATCONT has been improved and extended with many new functionalities:

1. The Select Cycle object is presented as a Special Point after each time integration. Selecting it as an initial point opens a subpanel with two fields that allow to choose a convergence criterion and a number of mesh intervals for initializing a curve of periodic orbits that starts from the periodic orbit found by time integration. The number of collocation points is always 4 by default. We note that this requires that the computed orbit covers between one and two periods of the periodic orbit.

2. Graphical 2D and 3D plots are reorganized: only one layout window is presented and nearly everything can be plotted, i.e. many earlier restrictions are lifted.

3. Clicking **Options**|**Plot Properties** in the main panel (Figure 1) opens an edit box panel which allows to assign plot properties (color, linestyle, ...) to each type of computed curve. Defaults are provided for all types of computed curves, which can be overwritten using MATLAB code in the edit boxes. For curves of equilibria and curves of limit cycles, it is possible to differentiate between stable and unstable parts of the curves. Advanced MATLAB users can edit the file *GUICurveModifications.m* to add new differentiations of the plot properties within a curve.

4. Error handling of plots is improved so that plot errors caused by command line interference, or by GUI interference

when computations are suspended, do not crash the computations.

5. Each input field has input restrictions and these are checked to minimize input errors. So for example it will not be possible to input a float or a question mark if a positive integer value is required. Errors are reported in the MATLAB command line. On the other hand, numerical fields can be filled with MATLAB expressions, provided they can be evaluated in the command line. So one can insert $2 * Pi$ instead of its decimal expansion $6.283184...$

6. In continuation plots it is possible to click on a found singular point to obtain information on the curve where it was found, the type of point and the normal form coefficients. By double clicking one selects the point as an initial point for another continuation.

7. The Scroll - key can be used to scroll through MATCONT windows. This functionality had to be implemented separately as MATLAB does not provide this functionality as part of their standard library.

8. Several special keys can be used to control continuation computations, namely: **Escape** to stop, **Space bar** to resume, **Enter** to pause and **Control** to continue (when pressed) or to pause (when released). The use of the **Control** key is new.

Example: Percept switching in the human visual system

Percept switching under the action of an ambiguous visual stimulus is a well-known phenomenon in psychophysics, see e.g. [5, 9]. In [9] the authors discuss a neural explanation of the stabilization of percept choices under intermittent viewing of an ambiguous stimulus. They consider the following system:

$$\begin{cases} X_1' &= (Stim - (1 + A_1)X_1 + \beta A_1 - \gamma S(X_2))/\tau \\ X_2' &= (Stim - (1 + A_2)X_2 + \beta A_2 - \gamma S(X_1))/\tau \\ A_1' &= -A_1 + \alpha S(X_1) \\ A_2' &= -A_2 + \alpha S(X_2) \end{cases} \tag{3}$$

with state variables X_1, X_2, A_1, A_2 and fixed parameters $\alpha = 5$, $\beta = 4/15$, $\gamma = 10/3$, and $\tau = 1/50$. The primary dynamical variables X_1, X_2 are the 'local fields', which correspond to the percept-related components of the membrane potentials of the neurons that encode the two competing percepts, indicated by 1 or 2 respectively. To each primary variable an adaptation variable is associated, called A_1, A_2 respectively. In the local field interpretation these correspond to the (averaged and scaled) gating variables of the neurons. $Stim$ is the amplitude of the stimulus. $S(X_1)$ is a sigmoidal function of X_1, zero for negative values of X_1 and equal to $X1^2/(1 + X1^2)$ for nonnegative values of X_1. $S(X_2)$ is to be interpreted similarly. The precise choice of the sigmoid function does not influence the qualitative behavior of (3).
Line 1 of (3) specifies how X_1 integrates the stimulus with its adaptation variable A_1 and the subtractive cross-inhibition $S(X_2)$. Lines 2 and 4 are dual to Lines 1 and 3.
In [9] the authors consider a 128 by 128 grid of points in a (T_{off}, T_{on}) space. For each point a simulation of (3) is done with the stimulus alternatingly switched off during a time span T_{off} and on during a time span T_{on}. The eventual behavior (after a transient) varies with the choice of T_{off} and T_{on} but also depends on the initial values of the state variables. It includes repeating and alternating patterns and bistability. An important observation is that for fixed T_{on} the behavior can be stabilized by increasing T_{off}, i.e. it leads to a situation where the percept is the same whenever the stimulus switches on (but may still depend on the initial state at the very beginning.)

Modelling of the percept switching system in MATCONT
In MATCONT we approximate the on/off switching by a continuous system with a periodic forcing with period $T_{off}+T_{on}$. This involves the inclusion of T_{off} and T_{on} as new parameters in the system and avoids the need of an enormous number of time integrations, in each of which the resulting stable behaviour can depend upon the initial state values.
Instead we make numerical computations based on the theory of bifurcations of periodic orbits. Boundaries of behavior regions in the (T_{off}, T_{on})-plane are obtained as curves of codimension 1 bifurcations of periodic orbits which meet in codimension 2 points. The periodic forcing is implemented with new independent state variables Y_1, Y_2. More precisely, we consider the following system (in the notation of the MATCONT input field):

```
S1=(X1^2)/(1+X1^2)/(1+exp(-expp*X1))
S2=(X2^2)/(1+X2^2)/(1+exp(-expp*X2))
omega=2*pi/(Ton+Toff)
Stim=1/(1+exp(-expp*(Y1-cos(2*pi*Ton/2/(Ton+Toff)))))
X1'=(Stim-(1+A1)*X1+beta*A1-gamma*S2)/tau
X2'=(Stim-(1+A2)*X2+beta*A2-gamma*S1)/tau
A1'=-A1+alpha*S1
A2'=-A2+alpha*S2
Y1'=-omega*Y2+Y1*(1-Y1^2-Y2^2)
Y2'=omega*Y1+Y2*(1-Y1^2-Y2^2)
```

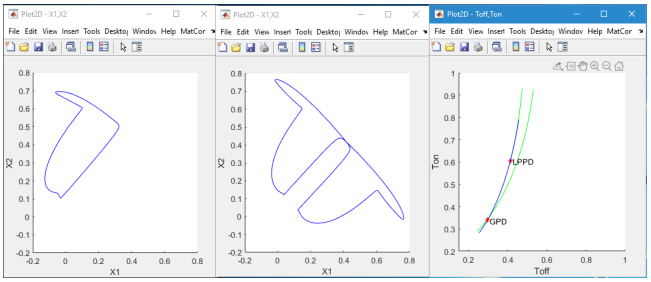

Figure 6: Left: repeating orbit with period $T_{off}+T_{on}$. Middle: alternating orbit with period $2(T_{off}+T_{on})$. Right: wedge of bistability.

with state variables X1, X2, A1, A2, Y1, Y2. The auxiliary variables S1,S2 approximate the sigmoid functions $S(X_1), S(X_2)$, respectively. They help to facilitate the implementation and increase the readability of the MATCONT input. We have also introduced an additional parameter $expp$ to the system, appearing in the equations for $S1$, $S2$ and $Stim$. By increasing its value, the analytical functions for S1 and S2 in the MATCONT system converge to the non-analytical step sigmoid functions that are used in (3). In our computations $expp = 60$. More details are given in [3].

The state variables Y1 and Y2 are decoupled from the other state variables. Their stable behaviour (after a transient) is a periodic orbit of the form $Y1 = \cos(\omega t + P), Y2 = \sin(\omega t + P)$ with a time shift P that depends only on the initial values of Y1,Y2. The period of this orbit is $T_{off} + T_{on}$.

When Y1,Y2 evolve along the unit circle then from the fourth line in the MATCONT system it follows that Stim is alternatingly close to one during a time span T_{on} and close to zero during a time span T_{off}. The stimulus term thus acts as an on/off switch in a periodic forcing of (3).

Fig. 6 (left) shows the (X1,X2) projection of a stable periodic orbit computed by time integration for $T_{off} = 0.6, T_{on} = 0.8$ (after a transient, starting with values of (Y1,Y2) which are not both zero). The time evolution is repeating, i.e. after each on/off cycle we see the same partial trajectory which corresponds to the percept X1 (for a different choice of the initial values of the state variables it can be the percept X2). Fig. 6 (middle) shows the projection of a stable periodic orbit computed by time integration for $T_{off} = 0.2, T_{on} = 0.8$. The time evolution is alternating, i.e. X1 and X2 dominate in turns.

To study the stability regions of repeating and alternating periodic orbits we use the Select Cycle functionality of MATCONT. We start the continuation of limit cycles from the stable limit cycles in Fig. 6 under variation of T_{off}. By using the branch switching functionalities of MATCONT we then construct the bifurcation diagram and find that in (T_{off}, T_{on})–space the stability regions of repeating and alternating orbits overlap in the wedge shown in Fig. 6 (right). The wedge is bounded at the right by a curve (green) of limit points of cycles of alternating periodic orbits. At the left, the boundary consists partly of a curve (green) of limit points of cycles of repeating periodic orbits and a curve (blue) of period-doubling points of repeating periodic orbits. The green and blue curves meet in a fold-flip bifurcation point of cycles denoted by LPPD. The blue curve contains a generalized period-doubling point (GPD) of repeating periodic orbits. The LPPD point is situated at $(0.41416, 0.60659)$, the GPD point at $(0.29837, 0.34146)$.

Conclusion

The MATCONT implementation greatly minimizes the computational work in the study of the perception problem. It confirms and explains the stabilization effect observed in the numerical simulations and allows easy extensions to similar studies with other models or other parameters.

References

[1] Dhooge, A. and Govaerts, W. and Kuznetsov, Yu. A. (2003) MATCONT: A MATLAB package for numerical bifurcation analysis of ODEs. *ACM Trans. Math. Softw.* **29(2)** pp. 141-164. http://doi.acm.org/10.1145/779359.779362.

[2] Dhooge A., Govaerts W., Kuznetsov Yu. A., Meijer, H.G.E., and Sautois, B. (2008) New features of the software MatCont for bifurcation analysis of dynamical systems. *Mathematical and Computer Modelling of Dynamical Systems* **14 (2),** pp. 147-175. https://doi.org/10.1080/13873950701742754

[3] Govaerts W., Kuznetsov Yu. A., Meijer, H.G.E., Neirynck, N. and van Wezel, R., Bistability and stabilization of human visual perception under ambiguous stimulation, *Nonlinear Dynamics, Psychology and Life Sciences*, **25 (3)** pp.297-307.

[4] Govaerts, W. and Kuznetsov, Yu.A. and Meijer, H.G.E. and Al-Hdaibat, B. and De Witte, V. and Dhooge, A. and Mestrom, W. and Neirynck, N. and Riet, A.M. and Sautois, B. (2019) MATCONT:Continuation toolbox for ODEs in Matlab. http://sourceforge.net/projects/matcont/

[5] Gregson, R.A.M. (2004) Transitions between two pictorial attractors. *Nonlinear Dynamics, Psychology and Life Sciences* **8, 41-64.**

[6] Khibnik, A.I. (1990) LINBLF: A program for continuation and bifurcation analysis of equilibria up to codimension three, in: Continuation and Bifurcation: Numerical Techniques and Applications, vol. 313 of NATO Adv. Sci. Inst. Ser. C, Math. Phys. Sci., Dordrecht (eds. Roose, D. and De Dier, B. and Spence, A.) pp.283-296. ISBN = 978-1-4020-6355-8.

[7] Kuznetsov, Yu. A. and Levitin, V.V. (1997) CONTENT: Integrated Environment for analysis of dynamical systems, CWI, Amsterdam.

[8] Neirynck, N. (2019). Advances in numerical bifurcation software: MatCont. *PhD thesis*, Ghent University, Belgium. https://biblio.ugent.be/publication/8615817

[9] Noest A. J., van Ee R., Nijs M. M., and van Wezel R. J. A. (2007) Percept-choice sequences driven by interrupted ambiguous stimuli: A low-level neural model. *Journal of Vision* **7,** pp. 1-14. https://doi.org/10.1167/7.8.10

[10] Roessler, O. (1979) Continuous chaos - four prototype equations, in: Bifurcation Theory and Applications in Scientific Disciplines (eds. Gurel, O. and Roessler, O.), New York Acad. Sci., pp. 376-392.