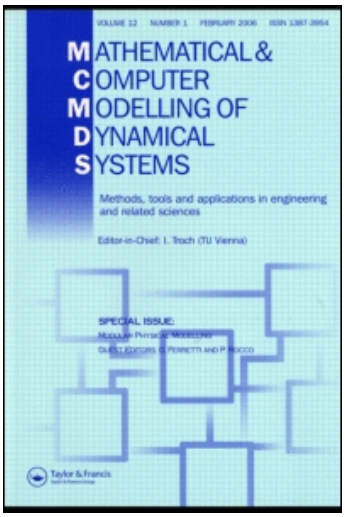

Mathematical and Computer Modelling of Dynamical Systems
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713682513

New features of the software MatCont for bifurcation analysis of dynamical systems

A. Dhooge [a]; W. Govaerts [b]; Yu. A. Kuznetsov [c]; H. G. E. Meijer [d]; B. Sautois [b]
[a] Department Informatietechnologie, Katholieke Hogeschool Sint-Lieven, Belgium [b] Department of Applied Mathematics and Computer Science, Ghent University, Belgium [c] Department of Mathematics, Utrecht University, The Netherlands [d] Department of Electrical Engineering, Mathematics and Computer Science, Twente Universtity, The Netherlands

Online Publication Date: 01 April 2008

To cite this Article 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,14:2,147 — 175
To link to this Article: DOI: 10.1080/13873950701742754
URL: http://dx.doi.org/10.1080/13873950701742754

Mathematical and Computer Modelling of Dynamical Systems
Vol. 14, No. 2, April 2008, 147–175

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

New features of the software MatCont for bifurcation analysis of dynamical systems

A. Dhooge[a], W. Govaerts[b]*, Yu.A. Kuznetsov[c], H.G.E. Meijer[d] and B. Sautois[b]

[a]*Department Informatietechnologie, Katholieke Hogeschool Sint-Lieven, Gebroeders Desmetstraat 1, B-9000 Gent, Belgium;* [b]*Department of Applied Mathematics and Computer Science, Ghent University, Krijgslan 281-S9, B-900 Gent, Belgium;* [c]*Department of Mathematics, Utrecht University, Budapestlaan 6, 3584 CD Utrecht, The Netherlands;* [d]*Department of Electrical Engineering, Mathematics and Computer Science, Twente Universtity, PO Box 217, 7500 AE Enschede, The Netherlands*

(*Received 14 March 2007; final version received 14 September 2007*)

Bifurcation software is an essential tool in the study of dynamical systems. From the beginning (the first packages were written in the 1970's) it was also used in the modelling process, in particular to determine the values of critical parameters. More recently, it is used in a systematic way in the design of dynamical models and to determine which parameters are relevant. MatCont and Cl_MatCont are freely available matlab numerical continuation packages for the interactive study of dynamical systems and bifurcations. MatCont is the GUI-version, Cl_MatCont is the command-line version. The work started in 2000 and the first publications appeared in 2003. Since that time many new functionalities were added. Some of these are fairly simple but were never before implemented in continuation codes, e.g. Poincaré maps. Others were only available as toolboxes that can be used by experts, e.g. continuation of homoclinic orbits. Several others were never implemented at all, such as periodic normal forms for codimension 1 bifurcations of limit cycles, normal forms for codimension 2 bifurcations of equilibria, detection of codimension 2 bifurcations of limit cycles, automatic computation of phase response curves and their derivatives, continuation of branch points of equilibria and limit cycles. New numerical algorithms for these computations have been published or will appear elsewhere; here we restrict to their software implementation. We further discuss software issues that are in practice important for many users, e.g. how to define a new system starting from an existing one, how to import and export data, system descriptions, and computed results.

Keywords: dynamical system; bifurcation; normal form; numerical continuation; MatCont

1. Introduction

Dynamical systems theory describes general patterns of solutions of ordinary differential equations (ODEs)

$$\frac{du}{dt} = f(u, \alpha), \quad u \in \mathbf{R}^n, \quad \alpha \in \mathbf{R}^{\mathrm{p}}. \tag{1}$$

Here f: $\mathbf{R}^n \times \mathbf{R}^p \to \mathbf{R}^n$ is assumed to be sufficiently smooth.

*Corresponding author. Email: willy.govaerts@ugent.be

ISSN 1387-3954 print/ISSN 1744-5051 online

DOI: 10.1080/13873950701742754
http://www.informaworld.com

The theory focuses on the change of the patterns under variation of parameter(s) α. Systems of ODEs have applications in many fields of research, including economics, engineering, biology, chemistry, and physics. In the rapidly expanding field of mathematical biology, we can refer to recent research in biochemistry [1], neuroscience [2], epidemiology [3], and immunology [4], among others.

The patterns identified by the theory are qualitative. Frequently the theory has been used to classify patterns rather than to build models that can be used for purposes of design or prediction. Computational capabilities have been a limiting factor in constructing such models since they rarely lend themselves to being solved with purely analytic methods. Attempts to apply the methods, developed by dynamical systems theory, to 'real world' problems has been a thoroughly interdisciplinary effort. For over twenty years, there has been a lively dialogue between mathematicians, scientists and engineers concerning the observation and interpretation of dynamical patterns in artificial and natural systems.

Traditionally, models of type (1) have been studied by simulating the time evolution $u(t)$ for a small range of different scenarios, i.e. initial points u_0 and parameter values α_0. In the last two decades, however, the focus has shifted towards classifying, as completely as possible, different types of behaviour that a dynamical system can exhibit as a function of its parameters. Particularly interesting is to study qualitative changes in the model behaviour under parameter variations (called *bifurcations*). A typical example of a bifurcation is a loss of stability by an equilibrium (constant solution), when one parameter varies. Theoretically analysed bifurcations include bifurcations near equilibria and cycles (periodic solutions), as well as bifurcations of manifolds composed of orbits which tend to equilibria and/or cycles as $t \to \pm\infty$ (see, for example, [5]). All bifurcations can be classified by their *codimension*, i.e. the number of control parameters one needs to tune in order to meet a bifurcation in a generic system (1). Numerical analysis of bifurcations requires special computations (e.g. continuation and normal form computations, construction of auxiliary maps and computing dimensional characteristics of attractors), which should be performed interactively. Therefore, bifurcation analysis should combine strong theoretical results, efficient numerical methods, and a user-friendly graphical interface.

From the beginning on, bifurcation software has also been used in the modelling proces. In its simplest form, this involves tuning the parameters of the system in such a way that the right bifurcations with the right properties (for example, a supercritical Hopf bifurcation) are found at the correct place, as a strong indication that the mathematical model behaves correctly. Examples of this strategy are given in classical modelling papers such as [6,7] and [8]. Not surprisingly, these papers are concerned with neuronal modelling, a field where the bifurcation structure is particularly important because neurons must be able to present a range of different behaviours and therefore tend to "sit" near bifurcation points.

More recently, bifurcation software is being used in a more systematic way to perform "inverse bifurcation analysis," where the basic idea is to map the space of bifurcation diagrams back to the space of parameters. Here one distinguishes the identification problem (which parameter configurations lead to a bifurcation diagram?) and the design problem (how do I choose parameters so that the system has a given bifurcation diagram?). These questions seem related but are in fact quite different: in the first case the nonuniqueness of the solution is a major problem, in the second it does not matter at all. A recent paper on the use of bifurcation software with good references to the literature is [9], a paper that focuses on gene regulatory networks.

The paper is organized as follows. In Section 2 we review the existing general-purpose tools for bifurcation analysis, and provide a survey of the provided functionalities. In Section 3 we give an overview of the features of MATCONT, including the Graphical User Interface and data handling. In Section 4 we discuss time integration and Poincaré maps. In Section 5 we review the computation of the phase response curve, a tool for sensitivity study of periodic orbits. In Section 6 we give details on the computation of normal form coefficients. Section 7 summarizes the functionalities of MATCONT for the continuation of branch points of equilibria and cycles. Section 8 describes the numerical continuation of homoclinic orbits. In Section 9 we briefly recall how C-code is introduced in MATCONT to improve performance.

MATCONT and CL_MATCONT are freely available at http://www.matcont.UGent.be upon registration. They require MATLAB 6.5 or higher. A manual and tutorials are also provided at the website.

2. Existing software

During the last decades, considerable efforts have been made to develop general-purpose software tools for bifurcation analysis (see Bibliographical Notes to Chapter 10 of [5] for references). One may distinguish at least three generations of such software:

(1) *Noninteractive packages and codes* were developed at the beginning of the 1980s and written in FORTRAN. They allowed one to continue equilibria and cycles of (1), as well as their simplest bifurcations (limit point, Hopf, period-doubling). The most widely used packages of this generation were AUTO86 and LINLBF. There were a few other programs available, which could perform normal form computations, namely: STUFF, BIFOR2, CYCLE.

(2) *Interactive programs* for bifurcation analysis of ODEs appeared at the end of 1980s, when workstations and IBM-PC compatible computers became widely available at universities and general research institutes. The first fully interactive bifurcation software was LOCBIF, that was based on LINLBF and worked on PCs under MS-DOS. It had a simple GUI with buttons, windows, and pull-down menus, characteristic for all programs of this generation. The programs AUTO94/97/2000 and XPPAUT, which ran under UNIX and used the efficient numerics of AUTO86, together with a simple GUI for X-Windows, also belong to this generation. All programs supported the on-line input of the right-hand sides of (1) and its compilation, either by a special built-in compiler or by calling a standard FORTRAN compiler. They allowed the user to continue equilibria, cycles, and their codimension-one bifurcations. Computed curves could be plotted in fixed graphic windows. All mentioned programs had a closed architecture.

(3) The third generation includes DSTOOL and CONTENT, developed in the 1990s. These programs have features of the so-called *software environments*, meaning that the user can define/modify a dynamical model (1), perform its rather complete analysis, and export results of this analysis in a graphical form, all without leaving the program. The programs have an elaborated GUI and run on several platforms, including popular workstations running Motif under UNIX, and PCs running OpenMotif under Linux or just MS-Windows. They provide off- or on-line help and extensive documentation for users and developers. It is possible, though hard, to extend them. Both programs support simulation of (1). DSTOOL computes equilibria and their codim 1 bifurcations using parts of the LINLBF-code. CONTENT,

that is written in C/C++, supports the continuation of equilibria of (1) and their bifurcations with codim ≤ 2, as well as the continuation of cycles using AUTO-like algorithms. Moreover, it computes normal forms for many equilibrium bifurcations, using internally generated symbolic derivatives of order ≤ 3.

Computations for ODEs are currently supported by the most widely used software packages AUTO97/2000 [10] and CONTENT 1.5 [11] are indicated in Table 1.

3. Bifurcation software MATCONT

Despite all efforts, none of the above mentioned packages covers the whole range of known bifurcations in ODEs (1), even with two control parameters ($p = 2$). The data exchange between existing programs is practically impossible due to individual data formats. Moreover, for some important bifurcation problems, such as normal form computations for cycle bifurcations, no robust and efficient numerical methods have been developed. None of the existing software represents the results of the analysis in a form suitable for standard control, identification, and visualization software or, therefore, fits well into the standard engineering software environments. Existing software tools are hardly extendible, since all of them are written in relatively low-level programming languages, like FORTRAN and C.

Recognizing these deficiencies lead to a research project to develop the next generation bifurcation software in MATLAB, now known as MATCONT, in which groups from Belgium and The Netherlands, as well as individual scientists from Canada, US, Switzerland, and Russia were involved. Early versions of MATCONT were described in

Table 1. Supported functionalities for ODEs in AUTO (A), CONTENT (C), and MATCONT (M).

	A	C	M
Time-integration		+	+
Poincaré maps			+
Monitoring user functions along curves computed by continuation	+	+	+
continuation of equilibria	+	+	+
Detection of branch points and codim 1 bifurcations (limit and Hopf points) of equilibria	+	+	+
Computation of normal forms for codim 1 bifurcations of equilibria		+	+
Continuation of codim 1 bifurcations of equilibria	+	+	+
Detection of codim 2 equilibrium bifurcations (cusp, Bogdanov-Takens, fold-Hopf, generalized and double Hopf)		+	+
Computation of normal forms for codim 2 bifurcations of equilibria			+
Continuation of codim 2 equilibrium bifurcations in three parameters		+	
continuation of limit cycles	+	+	+
Computation of phase response curves and their derivatives			+
Detection of branch points and codim 1 bifurcations (limit points, flip and Neimark-Sacker (torus)) of cycles	+	+	+
Continuation of codim 1 bifurcations of cycles	+		+
Branch switching at equilibrium and cycle bifurcations	+	+	+
Continuation of branch points of equilibria and cycles			+
Computation of normal forms for codim 1 bifurcations of cycles			+
Detection of codim 2 bifurcations of cycles			+
continuation of orbits homoclinic to equilibria	+		+

[12,13]. Here we focus on new capabilities implemented in the current version 2.2.* of MATCONT. We describe the software, including its design principles, as well as new and improved algorithms.

3.1. Overview of the new features

MATCONT 2.2.* has many new features and supports many functions that were not found in earlier packages. Solutions computable by and features present in the current version 2.2.* of MATCONT are indicated in the last column of Table 1.

It should be pointed out that MATCONT relies on the same numerical schemes for the detection of bifurcation points in parameter space and for continuation of bifurcation curves in parameter space as CONTENT. Most curves are computed by the same prediction-correction continuation algorithm based on the Moore-Penrose matrix pseudo-inverse. The continuation of bifurcation points of equilibria and limit cycles is based on bordering methods and minimally extended systems. Besides sophisticated numerical schemes, the software provides data storage and a modern graphical user interface (GUI). In addition, the package has an extensive on-line help for program use, and – more importantly – help on bifurcation analysis.

MATCONT supports time integration (simulation) and computes Poincaré maps of (1), i.e., it finds points of intersection of the orbits with a user-defined hypersurface in the state space.

The current implementation of these features in MATCONT is better than in CONTENT. In particular, MATCONT provides access to all standard ODE solvers supplied by MATLAB, as well as to two new stiff solvers, `ode78` and `ode87`. This makes MATCONT a unique simulation environment, since MATLAB itself does not have an interactive toolbox to run its solvers. The same representation of ODEs (1) is used in MATCONT as in MATLAB ODE solvers.

MATCONT is the first software, where the most efficient minimally extended defining systems, based on the bordering technique, are used for the two-parameter continuation of codim 1 bifurcations of equilibria and limit cycles of (1). In particular, the continuation of cycle bifurcations is also implemented using a variant of bordering for a boundary-value problem with an additional requirement concerning the rank defect of a differential operator [14,15].

MATCONT is the first software that computes phase response curves and their derivatives as a byproduct of the continuation of limit cycles. These curves are fundamental for study of the behaviour of oscillators and their synchronization and network properties.

The software also supports numerical normalization for all codim 1 cycle bifurcations. This allows one to compute critical coefficients of periodic normal forms [16] of restrictions of (1) to a centre manifold of the critical cycle, without computation of the Poincaré maps. This numerical method is very recent [17] and MATCONT is the only software that supports the normal form analysis of limit cycle bifurcations.

MATCONT incorporates another recent advance in the numerical bifurcation analysis: the continuation of branch points of equilibria and cycles [18]. It is the only standard software that can continue these points in three control parameters. For both equilibria and cycles, the implementation is based on the minimally extended defining systems using the bordering method. If artificial parameters are introduced, one can apply this implementation for the continuation of branch points in non-generic, i.e. symmetric or conservative systems, and in systems with invariant hyperplanes. It is remarkable that no

other software supports detection of branch points on curves of limit points, although they appear there generically.

Finally, the continuation of homoclinic orbits (both to hyperbolic saddles and to saddle-node equilibria) is now supported in MatCont as announced in [19], together with detection of a large number of codimension 2 bifurcations along the homoclinic curves.

It is envisaged to develop special versions of MatCont in the near future for large ODEs (1) and iterated maps.

3.2. *The graphs of adjacency*

Relationships between objects of codimension 0, 1 and 2 computed by MatCont 2.2.* are presented in Figures 1 and 2, while the symbols and their meaning are summarized in Tables 2 and 3, where the standard terminology is used (see [5]).

An arrow in Figure 1 from O to EP or LC means that by starting time integration from a given point we can converge to a stable equilibrium or a stable limit cycle, respectively. In general, an arrow from an object of type A to an object of type B means that that object of type B can be detected (either automatically or by inspecting the output) during the computation of a curve of objects of type A. For example, the arrows from EP to H, LP, and BP mean that we can detect H, LP and BP during the equilibrium continuation. Moreover, for each arrow traced in the reversed direction, i.e. from B to A, there is a possibility to start the computation of the solution of type A starting from a given object B. For example, starting from a BT point, one can initialize the continuation of both LP and H curves. Of course, each object of codim 0 and 1 can be continued in one or two system parameters, respectively.

The same interpretation applies to the arrows in Figure 2, where '*' stands for either S or U, depending on whether a stable or an unstable invariant manifold is involved.

In principle, the graphs presented in Figures 1 and 2 are connected. Indeed, it is known (see for example, [20,21]) that curves of codim 1 homoclinic bifurcations emanate from the BT, ZH, and HH codim 2 points. The current version of MatCont fully supports, however, only one such connection: BT → HHS.

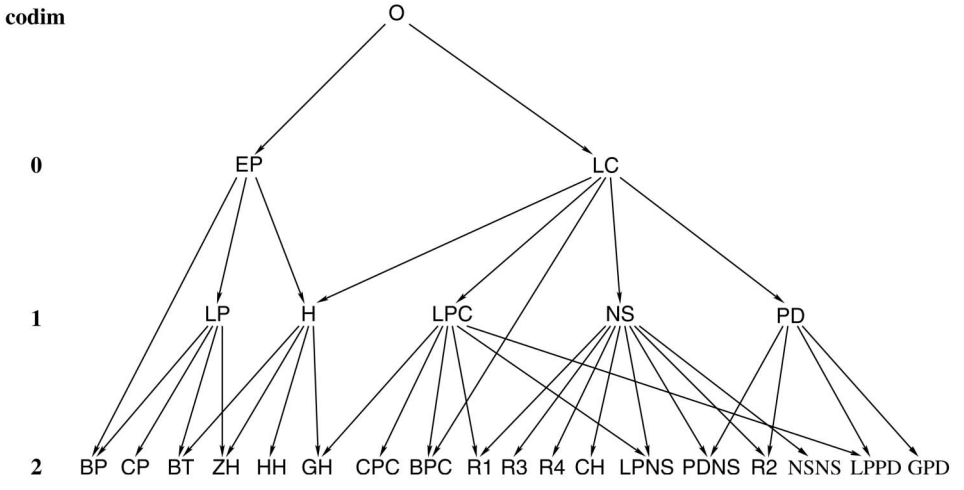

Figure 1. The graph of adjacency for equilibrium and limit cycle bifurcations in MatCont.

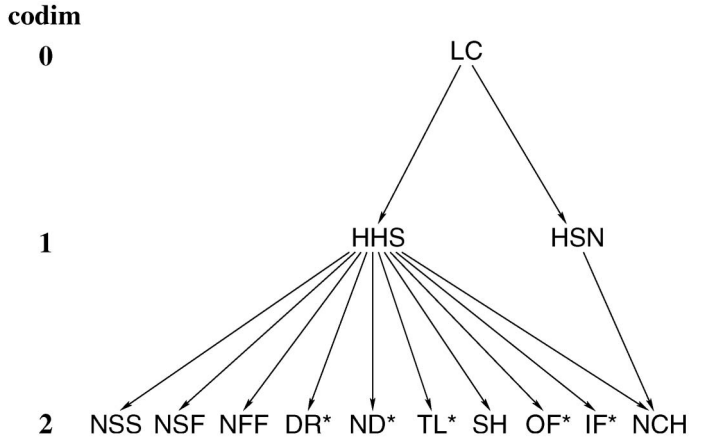

Figure 2. The graph of adjacency for homoclinic bifurcations in MATCONT; here * stands for S or U.

Table 2. Equilibrium- and cycle-related objects and their labels within the GUI.

Type of object	Label
Point	P
Orbit	O
Equilibrium	EP
Limit cycle	LC
Limit Point (fold) bifurcation	LP
Hopf bifurcation	H
Limit Point bifurcation of cycles	LPC
Neimark-Sacker (torus) bifurcation	NS
Period Doubling (flip) bifurcation	PD
Branch Point	BP
Cusp bifurcation	CP
Bogdanov-Takens bifurcation	BT
Zero-Hopf bifurcation	ZH
Double Hopf bifurcation	HH
Generalized Hopf (Bautin) bifurcation	GH
Branch Point of Cycles	BPC
Cusp bifurcation of Cycles	CPC
1:1 Resonance	R1
1:2 Resonance	R2
1:3 Resonance	R3
1:4 Resonance	R4
Chenciner (generalized Neimark-Sacker) bifurcation	CH
Fold – Neimark-Sacker bifurcation	LPNS
Flip – Neimark-Sacker bifurcation	PDNS
Fold-flip	LPPD
Double Neimark-Sacker	NSNS
Generalized Period Doubling	GPD

3.3. Continuation

CL_MATCONT forms a computational core of MATCONT, but can also be used independently as a general-purpose non-interactive continuation toolbox [13] in MATLAB.

Table 3. Objects related to homoclinics to equilibria and their labels within the GUI.

Type of object	Label
Limit cycle	LC
Homoclinic to Hyperbolic Saddle	HHS
Homoclinic to Saddle-Node	HSN
Neutral saddle	NSS
Neutral saddle-focus	NSF
Neutral Bi-Focus	NFF
Shilnikov-Hopf	SH
Double Real Stable leading eigenvalue	DRS
Double Real Unstable leading eigenvalue	DRU
Neutrally-Divergent saddle-focus (Stable)	NDS
Neutrally-Divergent saddle-focus (Unstable)	NDU
Three Leading eigenvalues (Stable)	TLS
Three Leading eigenvalues (Unstable)	TLU
Orbit-Flip with respect to the Stable manifold	OFS
Orbit-Flip with respect to the Unstable manifold	OFU
Inclination-Flip with respect to the Stable manifold	IFS
Inclination-Flip with respect to the Unstable manifold	IFU
Non-Central Homoclinic to saddle-node	NCH

It allows one to compute a sequence of consecutive points approximating a curve in $\mathbf{R}^N$ that is implicitly defined by

$$F(X) = 0, \quad F : \mathbf{R}^N \to \mathbf{R}^{N-1}. \tag{2}$$

The smooth map F is called the *defining function*. The continuer was developed to satisfy the following requirements:

(1) Universality, i.e. support of curves defined by various systems (2).
(2) Detection of singularities via curve-specific test functions.
(3) Singularity-specific locators.
(4) Processing of regular and singular points.
(5) Adaptation of defining systems and test functions.
(6) Support of user functions.
(7) Use of symbolic derivatives of F.
(8) Support of sparse matrices, e.g. F_X and its borderings.
(9) Providing a work space, i.e. handling temporary variables.

The continuer `cont.m` calls a `curve.m` file, which, however specific for each case, has a universal structure for all continuation problems. It includes the defining functions and their derivatives, test functions and rules for their application, as well as locators and processors of critical cases (singularities), and adapters of the defining and test functions. This idea was first implemented in CONTENT and proven to be useful. However, the implementation in CL_MATCONT is new. In particular the singularity-specific location and the possibility to provide curve-specific parameters for the continuer had not been implemented before. Support of sparse matrices is also a novelty.

3.4. Graphical user interface

MATCONT [12] is an interactive bifurcation environment in MATLAB that is based on CL_MATCONT. It supports the following actions:

(1) Specifying an initial solution to start the computation (either by selecting a previously computed solution or by on-line specification of a new solution).
(2) Selecting a solution type to compute.
(3) Setting and modifying of numerical parameters of the computation method.
(4) Activating detection of possible bifurcations.
(5) Defining user functions and activating the monitoring of their zeroes.
(6) Starting computation, allowing for termination and pausing.
(7) Supporting a database of computed curves and points.
(8) Controlling graphic and other output windows to represent the solutions during and after the computations.
(9) Context-dependent help.

MATCONT implements a general *Starter–Generator–Processor* technique to compute solutions (orbits and various bifurcation curves of (1)) that were also first tested in CONTENT. MATCONT deliberately mimics certain features of CONTENT's GUI to facilitate the migration of CONTENT users to the new environment, but the whole GUI implementation is new and incorporates several important improvements, including new design of Graphic Windows and a simplified manipulation with computed objects.

3.5. Importing and exporting systems and data

Systems and data can be imported into MATCONT and CL_MATCONT in various ways; output is provided in different forms.

Each system (1) is described by an m-file in the subdirectory Systems. For example, the description of a system named *Connor*, is stored in a file `Connor.m`. This file contains at least the description of the vector field, i.e. the right hand side $f(u, \alpha)$ of (1). Optionally, it can contain the symbolic description of the (partial) derivative tensors f_u, f_α, f_{uu}, $f_{u\alpha}$, f_{uuu}, f_{uuuu} and f_{uuuuu} (the last three are only used in computations of normal form coefficients). Finally, it can contain the description of any number of user functions, i.e. functions that can be monitored along computed curves and whose zeros can be detected and located. The system m-file can either be written by the user or generated automatically by the GUI of MATCONT. The latter is strongly encouraged if the Matlab symbolic toolbox is available and symbolic derivatives are desirable.

If MATCONT is used, then the most recent state of computations with the *Connor* system is stored in the `Connor.mat`-file in the same subdirectory Systems. This file stores the structures `FigPos` and `gds` which allow to restart the previous MATCONT session with `Connor`, even if computations with other systems were done since.

The essential numerical output of each continuation run `[x, v, s, h, f]` is produced by `cont.m`. Here the columns of `x` are the computed points and the columns of `v` are the tangent vectors. `s` is an array of structures each entry of which contains data related to a special point. Its first and last elements always refer to the first and last points of the curve, respectively, since for convenience these are also considered "special." `s` contains the fields `s.index`, `s.label`, `s.data` and `s.msg`. `s.index` is simply the array of indices of special points, so `s(1).index` is always equal to 1 and `s(end).index` is the number of

computed points. `s.label` is the label of the found special point and by convention `s(1).label` is "00" and `s(end).label` is "99". `s.data` is itself an array of structures. For each special point it contains fields with additional information, depending on the type of point. For example, there can be fields for normal form coefficients. In the case of branch points, it contains information that allows to distinguish the "old" branch on which the point was found from the "new" branch to which the user might wish to switch in another run. Finally `s.msg` is an array of strings that may contain any information which is useful for the user, for example the full name of the detected special point.

`h` contains some information on the continuation process. Its columns are related to the computed points. The first element of the *i*-th column is the stepsize used to compute the *i*-th point (zero for initial point and singular points). The second element is the number of Newton iterations used to compute that point (for singular points the number of locator iterations). The *i*-th column further contains the values of all active user functions and active test functions evaluated in the *i*-th computed point. This is very useful to check if, e.g. any bifurcation was left undetected because the locator did not converge.

The vector `f` is different, compared to older versions of MATCONT. For noncycle-related continuations, the `f`-vector just contains the eigenvalues, if asked for. For limit cycle continuations, it begins with the mesh points of the time- discretization, followed by, if they were asked for, the PRC- and dPRC-values in all points of the periodic orbit (see Section 5). Then, if required, follow the multipliers.

In the continuation of bifurcations of limit cycles, the time mesh is stored. Then follow the multipliers, if they were asked for (there is no meaningful phase response curve in these cases).

In MATCONT, all curves that have been computed using a specific system are stored in separate `.mat`-files, in a directory called *diagram*, under a subdirectory named after the system. For example, curves of the *Connor* system will be kept in `.mat`-files under the subdirectory `Systems/Connor/diagram/`. For continuation runs, each such `mat`-file contains the computed `x, v, s, h, f` arrays, plus the `cds` structure and a structure related to the curve type. Also, it contains the variables *point*, *ctype* and *num*. To understand their meaning, suppose that we are computing curves of limit cycles that we start from Hopf points. The first such computed curve then gets the name "H_LC(1)", *point* stores the string "H" and *ctype* stores the string "LC". Furthermore, *num* stores the index in `s` of the last selected point of the curve (the default is 1). The second curve of the same type is called "H_LC(2)" and so on. In fact, to save storage space, only a limited number of curves of a certain type is stored. This number can be set by the user and the default is 2. To save a computed curve permanently, the user must change its name.

For time integration runs, cf. Subsection 4.1 (Curve Type O) and Subsection 4.2.1 (Curve type DO), the `mat`-file contains `ctype`, `option`, `param`, `point`, `s`, `t`, `x`. Here `t` is the vector of time points and `x` is the corresponding array of computed points. `s` contains data on the first and last computed points. The meaning of `point, ctype` is similar to the case of continuation curves. Finally, `param` is the vector of parameters of the ODE (constant during time integration) and `option` is a structure that contains optional settings for time integration.

To export the computed results of a system to a different installation of MATCONT one has to copy the corresponding `m`-file, the `mat`-file and the directory of the system.

These files also contain all information needed to export the computed results to the general MATLAB environment, so MATCONT is really an open system.

MATCONT also produces graphical output. 2D and 3D graphs are plotted in MATLAB figure windows. Such a graph can be handled as any other graph that is produced in

MATLAB. It can be selected using the arrow-function of the MATLAB figure, and the line width, line style and colour can be altered. Markers can be set on the curve. It can be copy-pasted into another MATLAB figure. In a figure, textboxes can be inserted and axes labels can be added. Thus the user has a plethora of possibilities to combine different MATCONT output graphs into one figure.

Finally, we note that users often want to introduce new systems that are modifications of existing systems, but with slightly different sets of state variables and/or parameters. The best strategy to do this in MATCONT is first to edit the existing system, change its name to a new one and click "OK" to build an m-file with a different name and no associated directory of computed curves. Afterwards, one can edit the newly created system, make all desired changes and click "OK" again.

4. Time integration and Poincaré maps

MATLAB provides a suite of ODE-solvers. To create a combined integration-continuation environment, we have made them all accessible in MATCONT and added two new ones, `ode78` and `ode87`. `ode78` is an explicit Runge-Kutta method that uses 7th order Fehlberg formulas [22]. This is a 7-8th-order accurate integrator, therefore the local error normally expected is $O(h^9)$. `ode87` is a Runge-Kutta method that uses 8-7th order Dorman and Prince formulas. See [23]. This is a 8th-order accurate integrator therefore the local error normally expected is $O(h^9)$. In MATCONT the choice of the solver is made via the Integrator window that is opened automatically when the curve type O (Orbit) is chosen. We note that the Starter window has a SelectCycle button that allows to start the continuation of periodic orbits from curves computed by time integration.

4.1. Time integration

MATCONT uses the MATLAB representation of ODEs, which can also be used by the MATLAB ODE solvers. The user of CL_MATCONT is therefore also able to use his/her model within the standard ODE solver without rewriting the code. To make them accessible in MATCONT, some output functions and properties were created.

4.1.1. Solver output properties

The solver output properties allow one to control the output that the solvers generate. MATCONT detects whether extra output is needed by looking at the number of open output windows (2D, 3D or numeric). If extra output is required, MATCONT sets the OutputFcn property to the output function, `integplot` which is passed to an ODE solver by `options = odeset('OutputFcn', @integplot)`, otherwise it is set to the function `integ_prs`. The output function must be of the form `status = integplot (t, y, flag,` p_1`,` p_2`, . . .)`. The solver calls this function after every successful integration step with the following flags. Note that the syntax of the call differs with the flag. The function must respond appropriately:

- `init`: the solver calls `integplot (tspan, y0, 'init')` before beginning the integration, to allow the output function to initialize. This part initializes the output windows and does some initializations to speed up the further processing of the output. It also launches the window that makes it possible to interactively "stop/pause/resume" the computations.

- {none} : within MATCONT it is possible to define the number of points (*npoints*) after which output is needed. As the solver calls `status = integplot (t, y)` after each integration step, the number of calls to `integplot` and *npoints* does not correspond. Therefore this part is divided into two parts. `t` contains points where output was generated during the step, and `y` is the numerical solution at the points in `t`. If t is a vector, the *i*-th column of `y` corresponds to the *i*-th element of `t`. The output is produced according to the Refine option. `integplot` must return a status output value of 0 or 1. If *status* = 1, the solver halts integration. This part also handles the "stop/pause/resume" interactions.
- `done`: the solver calls `integplot([], [], 'done')` when integration is completed to allow the output function to perform any cleanup chores. The stop-window is also deleted.

Setting the OutputFcn property to the output function, `integ_prs`, reduces the output time. It only allows to interactively pause, resume and stop the integration.

4.1.2. *Jacobian matrices*

Stiff ODE solvers often execute the Jacobian matrix faster, i.e. the matrix of partial derivatives of the RHS that define the differential equations, is provided. The Jacobian matrix pertains only to those solvers for stiff problems (`ode15s, ode23s, ode23t, ode23tb`), for which it can be critical for reliability and efficiency. If you do not provide a function to calculate the Jacobian matrix, these solvers approximate it numerically using finite differences. Supplying an analytical Jacobian matrix often increases the speed and reliability of the solution for stiff problems. If the Jacobian matrix is provided (symbolically) in the odefile, MATCONT stores as property the function handle of the Jacobian, otherwise this handle is set empty.

The standard MATLAB `odeget` and `odeset` only support Jacobian matrices of the RHS w.r.t. the state variables. However, we do need derivatives with respect to the parameters for the continuation. To compute normal form coefficients, it is also useful to have higher-order symbolic derivatives. To overcome this problem, MATCONT contains new versions of `odeget` and `odeset`, which support Jacobian matrices with respect to parameters and higher-order partial derivatives w.r.t state variables. The new routines are compatible with the ones provided by MATLAB.

4.2. *Poincaré section and Poincaré map*

A Poincaré section is a surface in phase space that cuts across the flow of (1). It is a carefully chosen (in general, curved) surface in the phase space that is crossed by almost all orbits. It is a tool developed by Poincaré for visualization of the flow in more than two dimensions. The Poincaré section has one dimension less than the phase space and the Poincaré map transforms the Poincaré section onto itself by relating two consecutive intersection points, say u_k and u_{k+1}. We note that only those intersection points count, which come from the same side of the section. The Poincaré map is invertible because one gets u_n from u_{n+1} by following the orbit backwards. A Poincaré map turns a continuous-time dynamical system defined by (1) 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 is being attracted to a limit cycle, one observes dots converging to a fixed point in the Poincaré section.

4.2.1. Poincaré maps in MatCont

In some ODE problems the timing of specific events is important, such as the time and place at which a Poincaré section is crossed. For this purpose, MatCont provides the special curve type DO (Discrete Orbit) which is similar to O (Orbit) but provides a Starter window in which one can specify several event functions of the form $G(u) = \sum_{j=1}^{n} a_j u_j - a_0$, so $G(u) = 0$ corresponds to a Poincaré section (to avoid nonlinear interpolation problems, only this form of function G is allowed). While integrating, the ODE solvers can detect such events by locating transitions to, from or through zeros of $G(u(t))$. One does this by setting the Events property to a function handle, e.g. `@events`, and creating a function `[value, isterminal, direction] = events (t, y)` and calling `[t, Y, TE, YE, IE] = solver (odefun, tspan, y0, options)`.

For the i-th event function:

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

Corresponding entries in `TE, YE`, and `IE` return, respectively, the time at which an event occurs, the solution at the time of the event, and the index i of the event function that vanishes.

Unfortunately this method can only be used in Cl_MatCont. MatCont needs some interaction between `(t, Y)` and `(TE, YE)`. This is not possible by using the Events property. The part that handles the Poincaré map is the OutputFcn property of the integrators which is now set to `integplotDO`. The user is able to enter the equation $G(u)$ in the Starter window. This output function is divided into three parts:

- The solver calls `integplotDO` (tspan, y0, 'init') before starting the integration, to initialize the output function. Here $G(u(t_0))$ is initialized. The same output initializations are done as in the function `integplot` (see Subsection 4.1.1).
- The solver calls `integplotDO (tspan, y0)`. We are looking for a value t^* for which $G(u(t^*)) = 0$. The function `integDOzero` calculates the value of $G(u(t_i))$ and looks for a sign change. If there is one, the function tries to locate it. Afterwards, some processing is done.
- The solver calls `integplotDO (tspan, y0, 'done')` when the integration is completed, to allow the output function to perform any cleanup chores.

4.2.2. The Steinmetz-Larter example

In the peroxidase-oxidase reaction a peroxidase catalyzes an aerobic oxidation:

$$2YH_2 + O_2 + 2H^+ \rightarrow 2YH^+ + 2H_2O,$$

where YH_2 ($NADH$) is a general electron donor. Under the proper conditions this reaction yields oscillations in the concentrations of YH_2, O_2 and various reaction intermediates.

Steinmetz and Larter [24] derived the following system of four coupled nonlinear differential rate equations:

$$\begin{cases} \dot{A} &= -k_1ABX - k_3ABY + k_7 - k_{-7}A, \\ \dot{B} &= -k_1ABX - k_3ABY + k_8, \\ \dot{X} &= k_1ABX - 2k_2X^2 + 2k_3ABY - k_4X + k_6, \\ \dot{Y} &= -k_3ABY + 2k_2X^2 - k_5Y, \end{cases} \tag{3}$$

where A, B, X, Y are state variables and k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, and k_{-7} are parameters. Although highly simplified, the model is able to reproduce the three modes of simple, chaotic, and bursting oscillations found in experiments. State and parameter values of a point on a NS cycle in (3) are given in Table 4. The normal form coefficient of the NS cycle (see Subsection 6.2) is $-1.406017e-006$. Since it is negative, we expect stable tori nearby.

If we start a time integration from this NS point, with a slightly supercritical parameter value, namely $k_7 = 0.7167$, then after a transient the time-series exhibits modulated oscillations with two frequencies near the original limit cycle (see Figure 3). This is a motion on a stable two-dimensional torus that arises from the Neimark-Sacker bifurcation.

State and parameter values of another NS point of (3) are given in Table 5. The corresponding normal form coefficient (see Subsection 6.2) is $-2.919895e-008$, so that this bifurcation should also generate a stable torus. The NS point can be used as a starting point for the computation of a discrete orbit in the Poincaré section (called DO in the GUI) for a slightly decreased value of k_7, e.g. $k_7 = 1.51$. The output is shown in Figures 4 and 5, in which a cross-section of a stable torus is visible.

5. The phase response curve

The phase response curve of a limit cycle, or PRC, is a curve, defined over the period of the cycle, that expresses, at each time of that period, the effect of a small input vector on the cycle. In experimental circumstances, this may correspond to injected current, to the addition of more chemical agents, etc. A positive value means that the current cycle is shortened in time, a negative value means that the period is prolonged.

The PRC, as it is generally computed, is exact for infinitesimally small input vectors. In practice the maximum norm of the input vector would depend on the needed accuracy and the values of the system's state variables.

The derivative phase response curve or dPRC also has some very important applications. For the concrete use of PRC and dPRC in synchronization studies in neural modelling, we refer to [25].

Table 4. Values of a point on the first NS- cycle in the Steinmetz-Larter model.

Variable	Value	Parameter	Value	Parameter	Value
A	1.8609653	k_1	0.1631021	k_5	1.104
B	25.678306	k_2	1250	k_6	0.001
X	0.010838258	k_3	0.046875	k_7	0.71643356
Y	0.094707061	k_4	20	k_8	0.5
				k_{-7}	0.1175

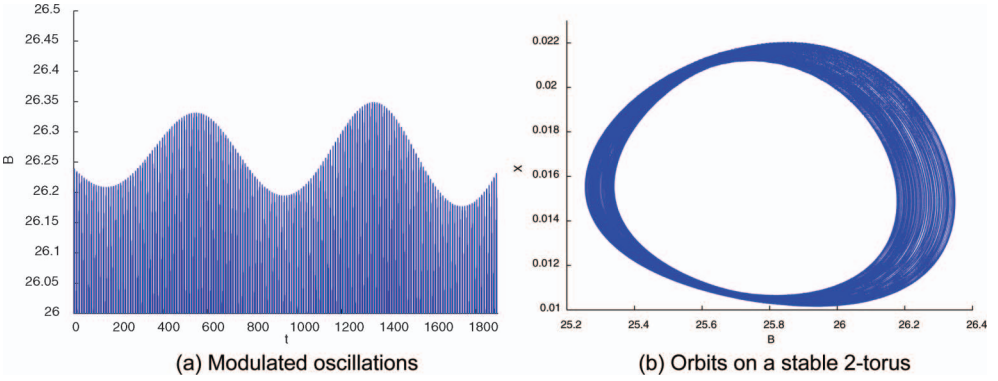

Figure 3. A stable torus in the Steinmetz-Larter example.

Table 5. Values of the second NS point in the Steinmetz-Larter model.

Variable	Value	Parameter	Value
A	6.1231735	k_7	1.5163129
B	9.1855407	k_8	0.83200664
X	0.0054271408		
Y	0.024602951		

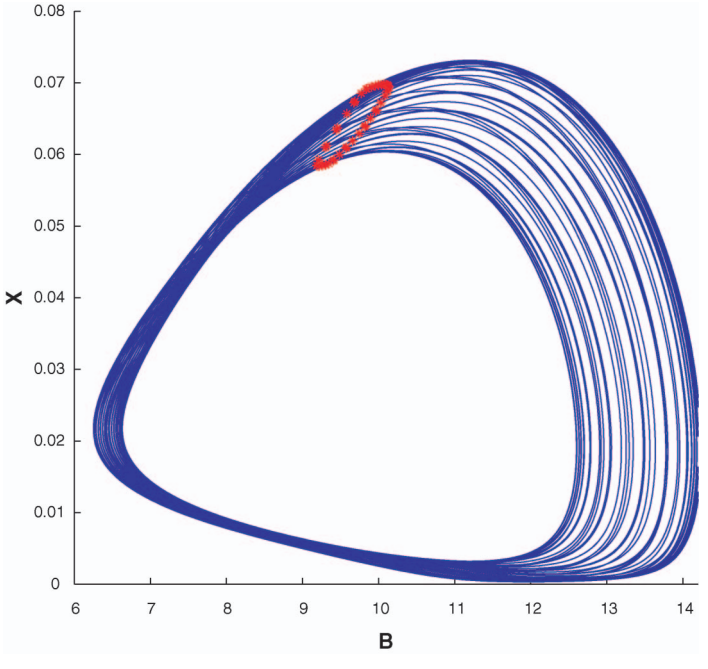

Figure 4. Dynamics on a stable torus.

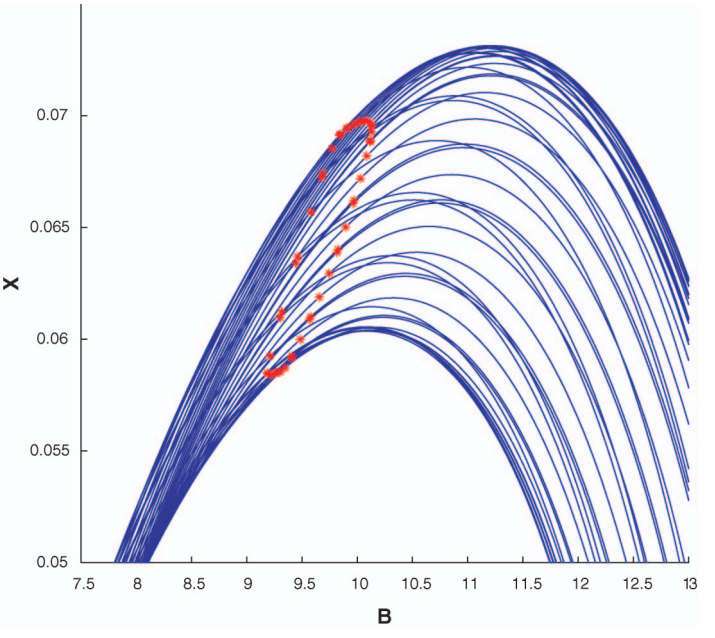

Figure 5. Poincaré map with a stable torus near the second NS bifurcation.

We have developed a new numerical method of computing the PRC and dPRC, that is specifically aimed at the computation during continuation of limit cycles. For the computation of one single PRC it is not less efficient than previous methods, but the advantage is less obvious. For details on this method, we refer to [26]. The standard method, which uses numerical integration of the adjoint system, was implemented in XPPAUT [27].

MatCont and Cl_MatCont support the computation of the PRC and dPRC of limit cycles during continuation, using this new method. The use in MatCont is easy: before starting the actual limit cycle continuation, the user can specify whether he wants to compute the PRC, dPRC or both, and he needs to indicate the input vector used. When a scalar is given as input, then the vector has this scalar as first entry and all other entries are zero. Then in separate plotting windows, for each computed step in limit cycle continuation, the PRC and/or dPRC are computed and plotted.

As an example application for the specific computation during continuation, one can study the evolution of the PRCs of limit cycles, when these approach certain bifurcations. For example the closer a limit cycle is to a homoclinic orbit, the smaller the negative part of the PRC becomes (relatively): it is harder to increase the period of the cycle. In Figure 6 two PRCs are shown for the same dynamical system, but at different distances from a homoclinic orbit. In Figure 7 a sequence of PRCs is shown, computed during the continuation of a limit cycle approaching a homoclinic orbit.

6. Computation of normal form coefficients

6.1. Numerical normal form coefficients for bifurcations of equilibria

While tracing an EP-curve, MatCont can detect two codim 1 bifurcations of equilibria, namely: LP (a Limit Point, where the equilibrium has eigenvalue $\lambda_1 = 0$) and H (a Hopf point, where the equilibrium has eigenvalues $\lambda_{1,2} = \pm i\omega_0$ with $\omega_0 > 0$), see Figure 1.

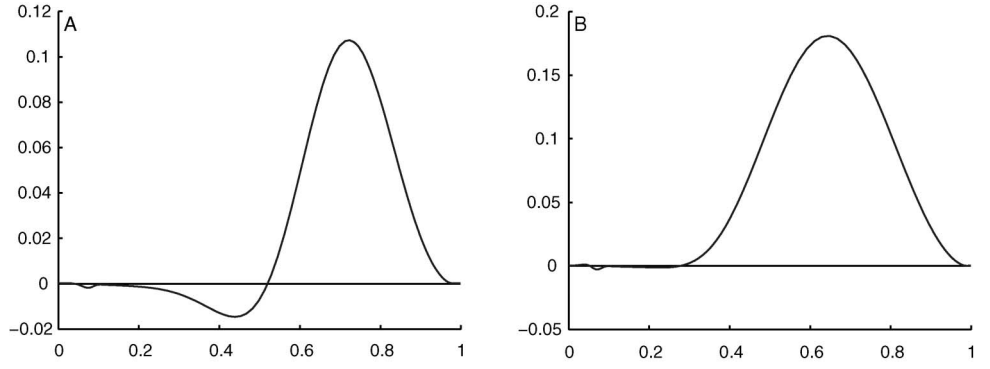

Figure 6. Two PRCs of limit cycles in the Morris-Lecar model [28], at different parameter values.

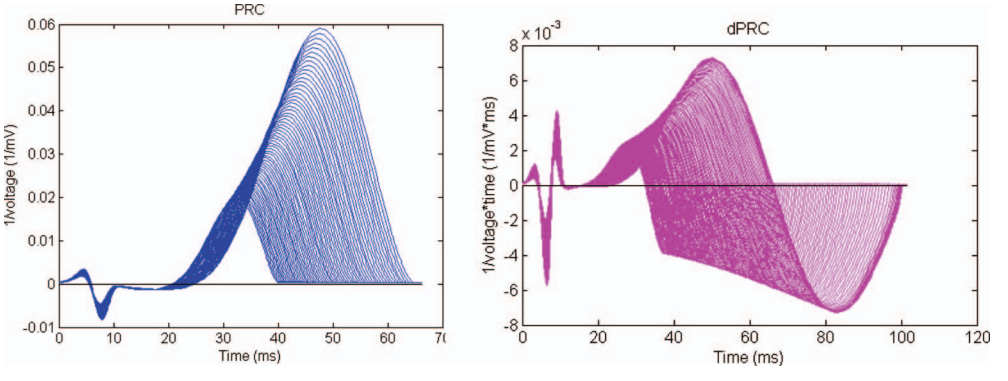

Figure 7. Left: PRCs of the Morris-Lecar system [28], computed during the limit cycle continuation. Right: the corresponding dPRCs.

The MatCont function `nf_LP.m` computes then the quadratic coefficient a of the restriction of (1) to its one-dimensional centre manifold at the LP-point. If $a \neq 0$, then two equilibria collide and disappear at this bifurcation generically. Another MatCont function – `nf_H.m` – computes the first Lyapunov coefficient l_1 of the restriction of (1) to its two-dimensional centre manifold at the H-point. When $l_1 < 0$, a stable (within the centre manifold) limit cycle appears; when $l_1 > 0$, an unstable limit cycle bifurcates.

To describe the computational algorithms, introduce

$$F(x) = f(u_0 + x, \alpha_0),$$

where u_0 is a bifurcating equilibrium at α_0, and write the Taylor expansion of F at $x = 0$ as

$$F(x) = Ax + \frac{1}{2}B(x,x) + \frac{1}{6}C(x,x,x) + \frac{1}{24}D(x,x,x,x) + \frac{1}{120}E(x,x,x,x,x) + O\left(\|x\|^6\right), \tag{4}$$

where $A = F_x(0) = f_u(u_0, \alpha_0)$ and the components of the multilinear functions B, C, D, and E are given by

$$B_i(u, v) = \sum_{j,k=1}^{n} \left.\frac{\partial^2 F_i(x)}{\partial x_i \partial x_k}\right|_{x=0} u_j v_k, \tag{5}$$

$$C_i(u, v, w) = \sum_{j,k,l=1}^{n} \left.\frac{\partial^3 F_i(x)}{\partial x_j \partial x_k \partial x_l}\right|_{x=0} u_j v_k w_l, \tag{6}$$

$$D_i(u, v, w, y) = \sum_{j,k,l,m=1}^{n} \left.\frac{\partial^4 F_i(x)}{\partial x_j \partial x_k \partial x_l \partial x_m}\right|_{x=0} u_j v_k w_l y_m, \tag{7}$$

and

$$E_i(u, v, w, y, z) = \sum_{j,k,l,m,s=1}^{n} \left.\frac{\partial^5 F_i(x)}{\partial x_j \partial x_k \partial x_l \partial x_m \partial x_s}\right|_{x=0} u_j v_k w_l y_m z_s, \tag{8}$$

for $i = 1,2,\ldots, n$.

The coefficient a at the LP-bifurcation is then given by the formula

$$a = \frac{1}{2}\langle p, B(q, q)\rangle, \tag{9}$$

where $Aq = A^{\mathrm{T}}p = 0$ and $\langle q,\ q\rangle = \langle p,\ q\rangle = 1$. The first Lyapunov coefficient l_1 at the H-bifurcation is computed via

$$l_1 = \frac{1}{2}\Re\Big(\Big\langle p, C(q, q, \bar{q}) - 2B\big(q, A^{-1}B(q, \bar{q})\big) + B\Big(\bar{q}, (2i\omega_0 I_n - A)^{-1}B(q, q)\Big)\Big\rangle\Big), \tag{10}$$

where $Aq = i\omega_0 q$, $A^{\mathrm{T}}p = -i\omega_0 p$, and $\langle p,\ q\rangle = 1$ (with an extra suitable normalization of q). The coefficients a and l_1 are used as test functions to detect and locate CP and GH codim 2 bifurcations, respectively, while the other codim 2 bifurcations are detected by testfunctions involving the Jacobian matrix only.

Thus, we need to evaluate the bilinear forms $B(q_1,q_2)$ and $C(q_1, q_1, q_2)$ on vectors q_1, $q_2 \in \mathbf{C}^{\mathrm{n}}$, as well as their scalar products

$$\langle p, B(q_1, q_2)\rangle, \langle p, C(q_1, q_1, q_2)\rangle$$

with another vector $p \in \mathbf{C}^{\mathrm{n}}$. If the MATLAB Symbolic toolbox is installed, symbolic derivatives of $f(u, \alpha)$ w.r.t. u can be used. In this case the calculation of tensor-vector products is straightforward.

Otherwise (or by user's request) numerical directional finite-differences are employed. Since

$$B(v, v) = \left.\frac{d^2}{d\tau^2}F(\tau v)\right|_{\tau=0}, \quad C(v, v, v) = \left.\frac{d^3}{d\tau^3}F(\tau v)\right|_{\tau=0}$$

can be used to compute the multilinear functions B and C with coinciding arguments, the symmetric differences in τ lead to the following approximations:

$$B(v, v) = \frac{1}{h^2}[F(hv) + F(-hv)] + O(h^2), \tag{11}$$

$$C(v, v, v) = \frac{1}{8h^3}[F(3hv) - 3F(hv) + 3F(-hv) - F(-3hv)] + O(h^2), \tag{12}$$

where $v \in \mathrm{C}^n$ and $h \ll 1$ is a small increment. Actually, the increments in (11) and (12) are selected individually to minimize the sum of the truncation and round-off errors. In MATCONT, the increment h_1 for the approximation of the Jacobian matrix A by central differences can be defined by the user. The increments of the kth-order derivatives are then computed according to $h_k = (h_1)^{3/(k+2)}$.

This allows us to approximate $B(q, q)$ and $C(q, q, q)$. However, as we already see from (10), the multilinear functions B, C, D and E have to be evaluated with non-coinciding arguments as well. For a general multilinear form $\omega(x) = \omega(x_1, x_2, \ldots, x_n)$ the *Sanders polarization identity* holds:

$$\omega(x) = \frac{1}{2^{n-1}n!}\sum_{i=1}^{2^{n-1}}\left[\prod_{j=2}^{n}(-1)^{\left\lfloor\frac{i}{2^{j-2}}\right\rfloor}\right]\omega\left(x_1 + \sum_{j=2}^{n}(-1)^{\left\lfloor\frac{i}{2^{j-2}}\right\rfloor}x_j, \ldots\right),$$

where $\lfloor a \rfloor$ is the largest integer k satisfying $k \le a$, while the dots stand for the repeating arguments. It implies the following identities

$$\begin{aligned}
B(u, v) &= \frac{1}{4}[B(u + v, u + v) - B(u - v, u - v)],\\
C(u, v, w) &= \frac{1}{24}[C(u + v + w, \cdots) - C(u + v - w, \cdots)\\
&\quad - C(u - v + w, \cdots) + C(u - v - w, \cdots)],\\
D(u, v, w, y) &= \frac{1}{192}[D(u + v + w + y, \cdots) - D(u + v + w - y, \cdots)\\
&\quad + D(u + v - w - y, \cdots) - D(u + v - w + y, \cdots)\\
&\quad - D(u - v + w + y, \cdots) + D(u - v + w - y, \cdots)\\
&\quad - D(u - v - w - y, \cdots) + D(u - v - w + y, \cdots)],
\end{aligned}$$

$$\begin{aligned}
E(u, v, w, y, z) = \frac{1}{1920}&[E(u + v + w + y + z, \cdots) - E(u + v + w + y - z, \cdots)\\
&+ E(u + v + w - y - z, \cdots) - E(u + v + w - y + z, \cdots)\\
&- E(u + v - w + y + z, \cdots) + E(u + v - w + y - z, \cdots)\\
&- E(u + v - w - y - z, \cdots) + E(u + v - w - y + z, \cdots)\\
&- E(u - v + w + y + z, \cdots) + E(u - v + w + y - z, \cdots)\\
&- E(u - v + w - y - z, \cdots) + E(u - v + w - y + z, \cdots)\\
&+ E(u - v - w + y + z, \cdots) - E(u - v - w + y - z, \cdots)\\
&+ E(u - v - w - y - z, \cdots) - E(u - v - w - y + z, \cdots)].
\end{aligned}$$

We use these formulas together with the standard finite-difference approximations of the fourth- and fifth-order directional derivatives, e.g.

$$D(v,\cdots) = \frac{1}{h^4}[F(2hv) - 4F(hv) - 4F(-hv) + F(-2hv)] + O(h^2),$$
$$E(v,\cdots) = \frac{1}{32h^5}[F(5hv) - 5F(3hv) + 10F(hv) - 10F(-hv) + 5F(-3hv) - F(-5hv)] + O(h^2),$$

to evaluate all needed forms and their scalar products with given vectors.

In general if two or more pairs of the vectors coincide, these formulas can be simplified. For example, we find the following identities:

$$B(v,w) = \frac{1}{4}[B(v+w,v+w) - B(v-w,v-w)],$$
$$C(v,v,w) = \frac{1}{6}[C(v+w,v+w,v+w) - C(v-w,v-w,v-w)] - \frac{1}{3}C(w,w,w),$$
$$D(v,v,w,w) = \frac{1}{12}[D(v+w,v+w,v+w,v+w) + D(v-w,v-w,v-w,v-w)] - \frac{1}{6}[D(v,v,v,v) + D(w,w,w,w)].$$

Thus we can reduce the approximation of $B(q_1, q_2)$ and $C(q_1, q_1, q_2)$ to that of the multilinear functions with coinciding arguments via (11) and (12). Moreover, not all terms of the polarization identity need to be computed, as several may coincide.

The current version of MATCONT also computes the critical normal form coefficients at all codim 2 bifurcations – CP, BT, GH, ZH, and HH – using (adapted) formulas from [29]. The normal form coefficients for codim 2 equilibrium bifurcations depend on kth-partial derivatives of the system's right-hand side at the critical point with $k \leq 5$. Similar to the above cases, their computation can be reduced to the evaluation of certain directional derivatives. This makes MATCONT the first software package that fully supports the normal form analysis of equilibrium bifurcations in two-parameter systems. Moreover, it computes necessary expansions of the unfolding parameters to switch to (all) limit cycle and (some) homoclinic bifurcation curves rooted there. The latter developments will be reported elsewhere.

6.2. *Numerical periodic normalization for bifurcations of limit cycles*

AUTO97/2000 and CONTENT 1.5 both support detection of codimension 1 bifurcations along one-parameter families of limit cycles, but normal forms are not computed. AUTO can continue all codimension 1 cycle bifurcations in two parameters. However, it provides no tools to detect and classify codimension 2 singularities. CONTENT computes the normal form coefficients at codimension 1 bifurcations of fixed points of maps and detects all eleven generic codimension 2 bifurcations. However, application of these capabilities of CONTENT to limit cycle analysis is usually based on the numerical construction of the

Poincaré map and the computation of its partial derivatives via finite differences, and, thus, unreliable.

MATCONT 2.2.* computes critical normal form coefficients for LPC, PD, and NS codim 1 bifurcations of limit cycles in (1) avoiding the numerical construction of the Poincaré map, via the periodic normalization method [17].

At an LPC-bifurcation, where a limit cycle has a non-semisimple Floquet multiplier $\mu_1 = 1$, MATCONT computes the quadratic coefficient b of the normal form of the restriction of (1) to the corresponding two-dimensional centre manifold. When $b \neq 0$, two limit cycles collide and disappear at the LPC-point. This normal form coefficient is computed by calling the function `nf_LPC.m` during the processing of the LPC-point; b is also used as a test function to detect a Cusp Point of Cycles (CPC).

At a PD-bifurcation, where the limit cycle has a multiplier $\mu_2 = -1$, MATCONT computes the cubic coefficient c of the normal form of the restriction of (1) to the corresponding two-dimensional centre manifold. If $c \neq 0$, then a limit cycle of (approximately) double period bifurcates from the original limit cycle at the PD-point, stable if $c < 0$ and unstable if $c > 0$. This normal form coefficient is computed by calling the function `nf_PD.m` during the processing of the PD-point; c is also used as a test function to detect a generalized period-doubling bifurcation (GPD).

At a NS-bifurcation, where the limit cycle has a pair of multipliers $\mu_{1,2} = e^{\pm i\theta}$, where $0 < \theta < \pi$ and $\theta \neq \frac{2\pi}{3}, \frac{\pi}{2}$, MATCONT computes the real part of the cubic coefficient d of the normal form of the restriction of (1) to the corresponding three-dimensional centre manifold. If $\Re(d) \neq 0$ then an invariant torus bifurcates from the original limit cycle, stable within the centre manifold if $\Re(d) < 0$ and unstable if $\Re(d) > 0$. This normal form coefficient d is computed by calling the function `nf_NS.m` during the processing of the NS; $\Re(d)$ is also a test function during the Neimark-Sacker continuation to detect a Chenciner bifurcation point (CH).

Several example computations of periodic normal form coefficients can be found in [17].

7. Continuation of branch points of equilibria and cycles

As we have seen, the singular points can be classified in terms of the codimension of the point type and the number of free parameters required to continue it, cf. Subsection 3.2.

Branch points disturb this nice picture. In principle, there are good reasons for not considering branch points at all in generic systems. However, problems arising in applications often have a special structure (e.g., equivariant, Hamiltonian, etc.). For this reason, standard software packages do provide the option to detect and accurately locate branch points, as well as branch switching, although they do not support their numerical continuation. One of the first standard codes that supported detection of equilibrium branch points and allowed branch switching was STAFF [30]. Similar facilities are provided by AUTO, CONTENT, and several other bifurcation programs.

Generic software can often deal with structured problems if an artificial "unfolding" parameter is introduced to break the special structure, thereby embedding the problem in a generic class; see, for example [31]. Thus, three-parameter continuation of branch points for generic systems is also useful for structured problems.

Therefore we have introduced the 3-parameter numerical continuation of branch points in MATCONT. The algorithms are based on bordered matrices for both detection and continuation, see [18] for details and examples.

7.1. *Continuation of branch points of equilibria*

During continuation of equilibria, MATCONT is able to detect branch points and locate them by using the specific BP Locator. The tangent vector at the singularity is computed within the locator. This is related to the processing of the branch point and is necessary to compute the direction of the secondary branch to switch branches. The parameter that was free during the equilibrium continuation becomes the branch parameter during the continuation of branch points.

MATCONT is also able to detect branch points during the continuation of limit points. To make this possible, we not only select two active parameters in the starter window but also select branch parameters. To specify active and branch parameters, a modification to the Starter window for the LP-continuation is done. In Figure 8 we present a screenshot of a MATCONT session in which a curve of limit points in a catalytic oscillator model was computed. The figure shows the MATCONT main window, the Starter window, the Continuer Window, and a 2D picture of a computed limit point curve on which two cusps (CP) and a Bogdanov-Takens point (BT) with normal form coefficients $(a, b) = (3.023978e-002, 2.133515e+000)$ as well as a branch point with respect to the third parameter q_3 are detected. From the Starter window one infers that the free parameters were q_5 and q_6; the detection of branch points with respect to q_3 was activated.

For each branch parameter that we select, a test function will be computed. The singularity matrix will automatically be extended. Those test functions can be monitored in the Numeric window. There is no need for a special branch point locator.

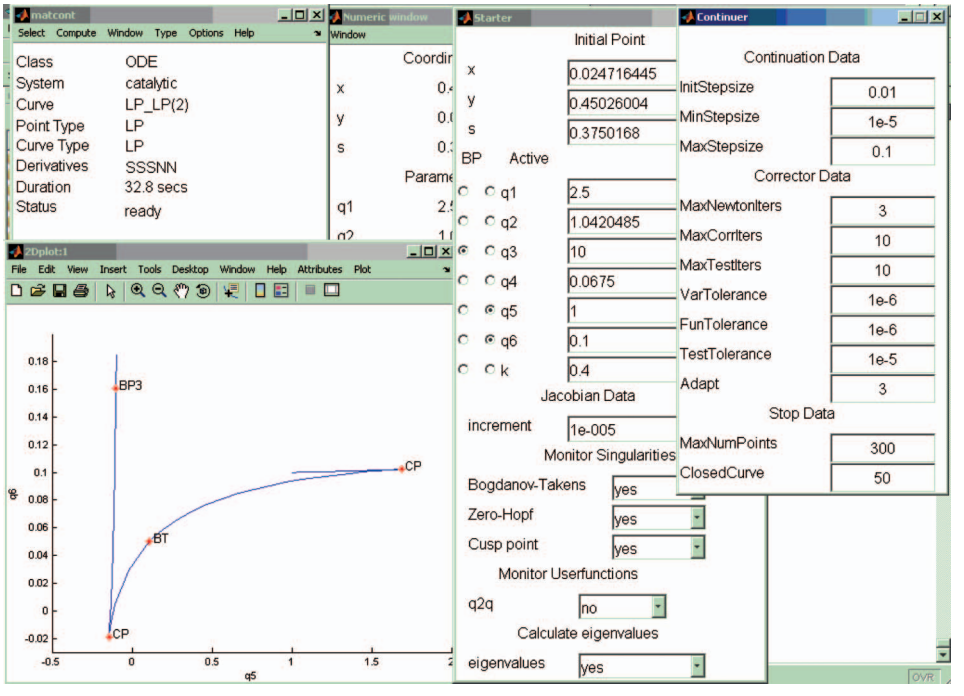

Figure 8. Screenshot of MatCont showing the main window, 2D-plot, starter and continuer windows in the case of continuation of an LP-curve with detection of CP, BT and BPs.

7.2. Continuation of branch points of limit cycles

During continuation of limit cycles, MATCONT 2.2.* is able to detect branch points of cycles (BPC's) and locate them using another specific branch point Locator. As in the equilibrium case, the tangent vector at the singularity is computed within the locator. This is related to the processing of the branch point and necessary to compute the direction of the secondary branch to make branch switching possible. The parameter that was free during the equilibrium continuation becomes the branch parameter during the continuation of branch points. MATCONT is also able to detect branch points during the continuation of LPC's.

We note that three example computations of branch points of cycles are given in [18] from different application fields, with and without equivariance properties.

8. Homoclinic orbits

8.1. Continuation of homoclinic orbits

In dynamical systems theory, an orbit corresponding to a solution $u(t)$ is called homoclinic to the equilibrium point u^0 of (1) if $u(t) \to u^0$ as $t \to \pm\infty$. There are two types of homoclinic orbits with codimension 1, namely homoclinic-to-hyperbolic-saddle (HHS), if u^0 is a saddle (saddle-focus or bi-focus), and homoclinic-to-saddle-node (HSN), if u^0 is a saddle-node (i.e., exhibits a limit point bifurcation). We recall that AUTO has a toolbox for homoclinic continuation, named HOMCONT [32,33].

MATCONT and CL_MATCONT allow one to continue both HHS and HSN orbits. These continuations can be initiated either from a Bogdanov-Takens (BT) point (using the method by [34], see also [35]) or from a limit cycle of a large period. For a homoclinic continuation, there are three so-called homoclinic parameters: `T`, `eps0` and `eps1`. The part of the infinite cycle that is actually stored and computed for each orbit has length `2T`; `eps0` and `eps1` are the distances of the equilibrium to the (respectively) first and last point of the stored part of the orbit.

To start a homoclinic continuation from a limit cycle with large period or from a previously computed homoclinic orbit, the user must declare that the point is a homoclinic orbit. Automatically, initial values for the homoclinic parameters will be computed. Then the user has to select 2 free system parameters, and 1 or 2 of the homoclinic parameters. The size of the system will automatically be adjusted to the choice of the user. An example Starter window is shown in the left of Figure 9.

To start from a BT point, the user only needs to define a value for the initial amplitude. This is the amplitude of the first computed homoclinic orbit, usually this should not be too small (e.g. 0.1 or 0.01). The user also needs to select two free system parameters. By default, T and eps0 are the free homoclinic parameters. An example Starter window is shown in the right of Figure 9.

During the continuation, it is necessary to keep track of several eigenspaces of the equilibrium in each step. To do this in an efficient way, MATCONT incorporates the continuation of invariant subspaces [36] into the defining system. For some details on the implementation of the homoclinic continuation we refer to [19].

During the continuations, detection of a large number of codimension two bifurcations is supported. They are summarized in Subsection 3.2, (see Table 3 and Figure 2). The test functions for these bifurcations were adopted from [33], except for the inclination-flip bifurcations. For these a more efficient method was developed, such that no adjoint variational equation is solved separately, see [19].

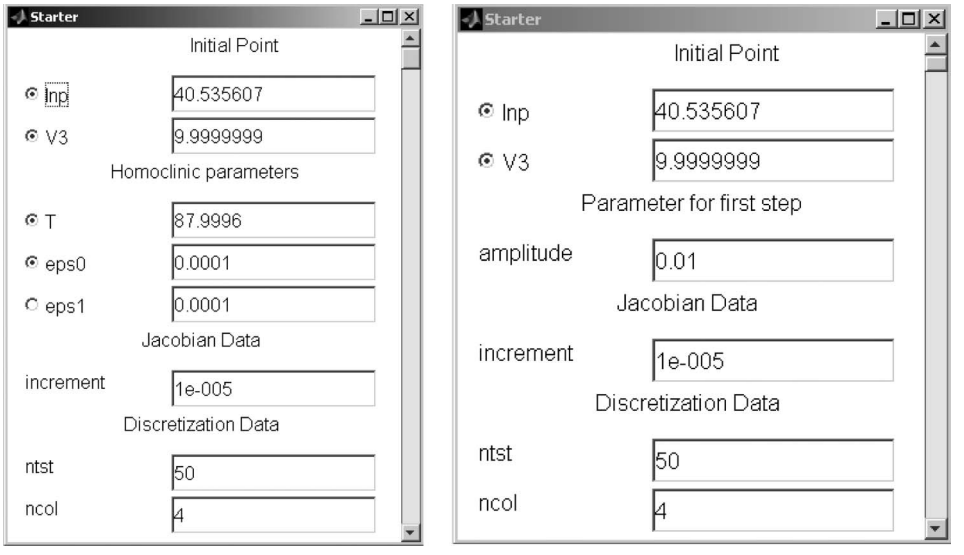

Figure 9. (Parts of) the starter windows when continuing a curve of HHS orbits, starting from a limit cycle or homoclinic orbit (left) or from a BT point (right).

8.2. *The Koper example*

Consider the following system of differential equations:

$$\begin{cases} \epsilon_1 \dot{x} &= (k\ y - x^3 + 3x - \lambda) \\ \dot{y} &= x - 2y + z \\ \dot{z} &= \epsilon_2 (y - z). \end{cases} \tag{13}$$

This system, which is a three-dimensional van der Pol-Duffing oscillator, has been introduced and studied by [37]. It is used as a standard demo in HOM-CONT. Parameters ϵ_1 and ϵ_2 are kept at 0.1 and 1, respectively. We note that system (13) has certain symmetry: If $(x(t),\ y(t),\ z(t))$ is a solution for a given value of λ, then $(-x(t),\ -y(t),\ -z(t))$ is a solution for $-\lambda$.

Starting from a general point $(0,\ -1,\ 0.1)$ and setting $k = 0.15$ and $\lambda = 0$, we find by time integration a stable equilibrium at $(-1.775,\ -1.775,\ -1.775)$.

By equilibrium continuation with λ free, we find two limit points (LP), at $(-1.024695,\ -1.024695,\ -1.024695)$ for $\lambda = -2.151860$ with normal form coefficient $a = -4.437056e+000$ and at $(1.024695,\ 1.024695,\ 1.024695)$ for $\lambda = 2.151860$ with normal form coefficient $a = -4.437060e+000$ (note the reflection).

By continuation of the limit points with (k, λ) free, MATCONT detects a cusp point CP at (0,0,0) for $k = -3$ and $\lambda = 0$. Also detected are two Zero-Hopf points ZH for $k = -0.3$ at $\pm(0.948683,\ 0.948683,\ 0.948683)$ and $\lambda = \pm 1.707630$, but these are in fact Neutral Saddles. Further, two Bogdanov-Takens points BT are found for $k = -0.05$ at $\pm(0.991632,\ 0.991632,\ 0.991632)$ and $\lambda = \pm 1.950209$. The normal form coefficients are $(a,\ b) = (6.870226e+000,\ 3.572517e+001)$.

A bifurcation diagram of (13) is shown in Figure 10.

We now compute a HHS curve starting at one of the BT points, and setting the curve-type to Homoclinic. To further initialize the homoclinic continuation, some parameters have to be set and/or selected in the Starter window.

First we select two system parameters as free parameters, e.g. k and λ. We must also set the *Initial amplitude*, i.e. an approxinate size of the first homoclinic orbit. The default setting of 1e-2 is a good value in most cases.

Another option is to start from a limit cycle with large period. For example, select the BT point at $\lambda = 1.950209$, and then compute a curve of Hopf points H passing through it, along which one encounters a Generalized Hopf bifurcation GH. Stop the continuation at (the random value) $\lambda = 1.7720581$, where the Hopf point is at (0.98526071, 0.98526071, 0.98526071) and $k = -0.23069361$. We then can find limit cycles for very slowly decreasing values of λ, (λ decreases down to 1.77178), with a rapid increase in the period. At some point, one can stop the continuation, and switch to the continuation of the homoclinic orbit (Figure 11).

One can also monitor the eigenvalues of the equilibrium during continuation, by displaying them in the Numerical window. This is a very useful feature, because it gives

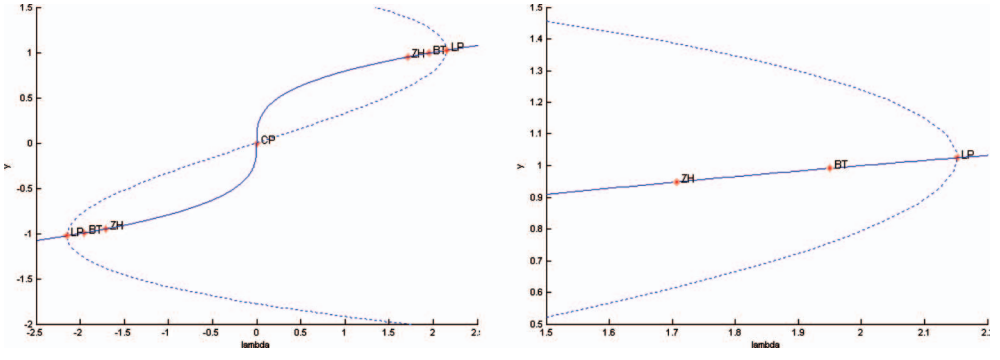

Figure 10. Left: equilibrium bifurcation diagram of the Koper system. Dashed line = equilibria, full line = limit points, CP = Cusp Point, BT = Bogdanov-Takens, ZH = Zero-Hopf, LP = Limit Point. Right: zoom on the left part of the diagram.

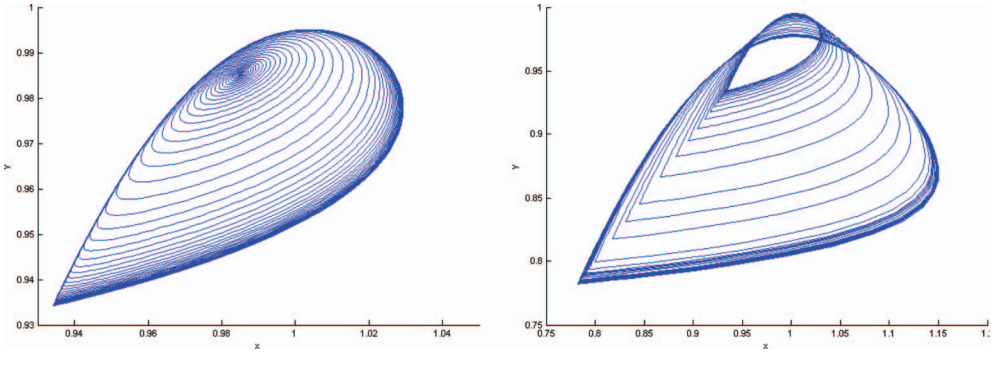

Figure 11. Left: limit cycles, starting from a Hopf point H and approaching a homoclinic orbit. Right: a family of HHS orbits.

indications on what further bifurcations might be expected. For example, a non-central homoclinic-to-saddle-node reveals itself by the fact that one eigenvalue approaches zero. Once one is close enough to such a point, the user can switch to the continuation of HSN orbits from there.

9. Improving performance

9.1. *Incorporating C-code into Matlab*

To speed up the computations, we must step away from the idea of writing everything in the MATLAB-language. The basic version of MATLAB contains its own C-compiler. It is possible, to let the C-code and MATLAB-code communicate with each other.

As the compiler is present in every version of MATLAB, we can supply the C-code and let the program compile the C-code at the startup of MATCONT. This causes a slight loss of time at that particular moment, negligible when compared to the time gained when continuing limit cycles and their bifurcations. The advantage of this approach is that it keeps the program completely platform-independent. The compiled C-code, which is platform-dependent, is only generated at runtime, and is thus adapted to the machine.

The compiled function can be called from the MATLAB-code, as if it were a MATLAB-function. We note that C-code written to communicate with a MATLAB-program, is not a pure C-code. A standard c-compiler would object to parts of that code because of some intermediate classes of variables, and intermediate types of commands, which are not defined in standard C. There is for example the type `MxArray`, which is the standard datatype to pass parameter lists between a C-code and MATLAB. Memory allocation happens through special commands as `MxCreateDoubleMatrix` and `MxCreateSparse`.

The key is in the parameter lists. The C-code receives all parameters from MATLAB in one big object of the type pointer to `MxArray`, and also passes all return variables back to the calling MATLAB-code in such a construct. The difficulty when writing the special C-code is to analyse that object, to decide what kinds of parameters are actually passed, and to store them in the C-variables before doing any real computations in C. Analogously, before closing the C-function, all return-variables must be stored in a pointer to `MxArray`, in such a way that the calling MATLAB-function will be able to read and recognize all variables.

In the C-code, it is sometimes necessary or just better to call a MATLAB-function or operator. For example, to solve a big linear system it can be faster to use the backslash operator in MATLAB than to use own C-code. And again, there is a solution. As it is possible to call compiled C-functions from MATLAB, it is also possible to call MATLAB operators or self-written MATLAB-functions from C, as long as the C-code makes are sure that all parameters are stored properly, before passing them on to the MATLAB-code, and that the C-code processes the return values correctly.

9.2. *C-code in MATCONT*

In the pure MATLAB version of CL_MATCONT, over half the time spent in the continuation of limit cycles, is actually spent in the evaluation of the Jacobian matrix of the discretized boundary-value problem implemented in `BVP_LC jac.m`.

Thus, this function is a natural choice for re-implementation in C. Other matrix evaluations implemented in MATCONT 2.2.* in C are similar to `BVP_LC_jac.m`:

- `BVP_PD_jac.m`: used in the continuation of PD cycles,
- `BVP_BPC_jacC.m`: used in the location of BPC cycles,

- `BVP_BPC_jacCC.m`: used in the continuation of BPC cycles,
- `BVP_LPC_jac.m`: used in the continuation of LPC cycles,
- `BVP_NS_jac.m`: user in the continuation of NS cycles.

This results in a significant gain in efficiency. In Tables 6 and 7, it is shown that the relative improvement by using the C-code is around 25% when turning off bifurcation detection, and over 50% when the test functions are turned on.

In the same tables, a timing comparison with the CONTENT-package [11] is shown under the identical GUI-load. The *adapt* system is a very simple model from adaptive control considered as a test example in [12]. The *Connor* system [38] is a complicated neural model with six state variables. All computations were done under Windows XP on a Intel (R) Pentium (R)M computer with a 1.73 GHz computer with clock speed 795 MHz.

It is clear that on level of timing a MATLAB-program, even when using the C-code, cannot compete with the fully compiled software CONTENT. However, by using MATLAB a lot can be gained in ease of installation, user-friendliness, extendability and functionalities in general. Also conversion of data is an important issue, and many scientists already know MATLAB, which lowers the threshold to start using MATCONT. Moreover, the C-reimplementation of additional functions of MATCONT can be done independently to further improve the performance of MATCONT without disabling any of its features.

Table 6. Timing results for limit cycle-continuation, for 200 limit cycles in the adapt-system.

Mesh intervals	Testfunctions	MATCONT 2.2.* (C-code)	MATCONT 1.1 (M-code)	Improvement	CONTENT
50	off	32 s	44 s	27%	6 s
50	on	53 s	121 s	56%	7 s
100	off	86 s	115 s	25%	11 s
100	on	145 s	340 s	57%	13 s
200	off	265 s	339 s	22%	14 s
200	on	464 s	1077 s	57%	20 s

[a]The first column states the number of mesh intervals. The second column mentions whether the testfunctions for bifurcations and the multipliers were turned on. The third, fourth and fifth column give timing results for the newest version of MATCONT, for the previous (full Matlab) version of MATCONT, the relative improvement between versions, and the timing result for CONTENT.

Table 7. Timing results for limit cycle continuation, for 100 limit cycles in the Connor-system.

Mesh intervals	Testfunctions	MATCONT 2.2.* (C-code)	MATCONT 1.1 (M-code)	Improvement	CONTENT
50	off	45 s	55 s	18%	7 s
50	on	78 s	170 s	54%	11 s
100	off	133 s	169 s	21%	13 s
100	on	250 s	564 s	56%	22 s

[a]The first column states the number of mesh intervals. The second column mentions whether the testfunctions for bifurcations and the multipliers were turned on. The third, fourth and fifth column give timing results for the newest version of MATCONT, for the previous (full Matlab) version of MATCONT, the relative improvement between versions, and the timing result for CONTENT.

10. Conclusions

We discussed the numerical continuation packages MATCONT and CL_MATCONT for the interactive study of dynamical systems and bifurcations, in particular the recently added new features. This includes Poincaré maps, continuation of homoclinic orbits, periodic normal forms for codimension 1 bifurcations of limit cycles, normal forms for codimension 2 bifurcations of equilibria, detection of codimension 2 bifurcations of limit cycles, automatic computation of phase response curves and their derivatives, continuation of branch points of equilibria and limit cycles.

We further discussed software issues that are in practice important for many users, e.g. how to define a new system starting from an existing one, how to import and export data, system descriptions, and computed results.

MATCONT and CL_MATCONT have already often been used in the modelling process, in particular in biological and biochemical modelling. We expect that with the new features they will be even more useful to many users.

Acknowledgements

Bart Sautois is a research assistant of the Research Foundation – Flanders (FWO - Vlaanderen).

References

[1] A. Beuter et al. *Nonlinear Dynamics in Physiology and Medicine*, Springer, New York, 2003. Interdisciplinary Applied Mathematics, Vol. 25.

[2] F.C. Hoppensteadt and E.M. Izhikevich, *Weakly Connected Neural Networks*, Springer-Verlag, New York, 1997.

[3] O. Diekmann and J.A.P. Heesterbeek, *Mathematical Epidemiology of Infectious Diseases: Model Building, Analysis and Interpretation*, Wiley, New York, 2000. Wiley Series in Mathematical and Computational Biology.

[4] M.A. Nowak and R.M. May, *Virus Dynamics: Mathematical Principles of Immunology and Virology*, Oxford University Press, Oxford, 2000.

[5] Yu.A. Kuznetsov, *Elements of Applied Bifurcation Theory*, 3rd ed., Springer, New York, 2004.

[6] J. Rinzel and Y.S. Lee, *Dissection of a model for neuronal parabolic bursting*, J. Math. Biol. 25 (1987), pp. 653–675.

[7] J. Guckenheimer, S. Gueron, and R.M. Harris-Warrick, *Mapping the dynamics of a bursting neuron*, Phil. Trans. R. Soc. Lond. B 341 (1993), pp. 345–359.

[8] R. Bertram, M.J. Butte, T. Kiemel, and A. Sherman, *Topological and phenomenological classification of bursting oscillations*, Bull. Math. Biol. 57 (1995), pp. 413–439.

[9] J. Lu, H.W. Engl, and P. Schuster, *Inverse bifurcation analysis: application to simple gene systems*, Algorithms Mol. Biol. 1 (2006), pp. 1–16.

[10] E.J. Doedel, A.R. Champneys, T.F. Fairgrieve, Yu.A. Kuznetsov, B. Sandstede, and X.J. Wang, AUTO97-AUTO2000, *Continuation and bifurcation software for ordinary differential equations (with HomCont). User's guide*, 1997–2000. Available online at: http://indy.cs.concordia.ca

[11] Yu.A. Kuznetsov and V.V. Levitin, *CONTENT: integrated environment for analysis of dynamical systems*, 1997. Available online at: ftp://ftp.cwi.nl/pub/CONTENT

[12] A. Dhooge, W. Govaerts, and Yu.A. Kuznetsov, *MATCONT: a Matlab package for numerical bifurcation analysis of ODEs*, ACM Toms 29 (2003), pp. 141–164.

[13] A. Dhooge, W. Govaerts, Yu.A. Kuznetsov, W. Mestrom, and A.M. Riet, *CL_MATCONT: a continuation toolbox in Matlab*. Proceedings of the 2003 ACM Symposium on Applied Computing, Melbourne, FloridaUSA, 2003 pp. 161–166.

[14] E.J. Doedel, W. Govaerts, and Yu.A. Kuznetsov, *Computation of periodic solution bifurcations in ODEs using bordered systems*, SIAM J. Numer. Anal. 41 (2003), pp. 401–435.
[15] W. Govaerts, Yu.A. Kuznetsov, and A. Dhooge, *Numerical continuation of bifurcations of limit cycles in Matlab*, SIAM J. Sci. Comput. 27 (2005), pp. 231–252.
[16] G. Iooss, *Global characterization of the normal form for a vector field near a closed orbit*, J. Diff. Eqs. 76 (1988), pp. 47–76.
[17] Yu.A. Kuznetsov, W. Govaerts, E.J. Doedel, and A. Dhooge, *Numerical periodic normalization for codim 1 bifurcations of limit cycles*, SIAM J. Numer. Anal. 43 (2005), pp. 1407–1435.
[18] E.J. Doedel, W. Govaerts, Yu.A. Kuznetsov, and A. Dhooge, *Numerical continuation of branch points of equilibria and periodic orbits*, Int. J. Bifurcat. Chaos 15 (2005), pp. 841–860.
[19] M. Friedman, W. Govaerts, Yu.A. Kuznetsov, and B. Sautois, *Continuation of homoclinic orbits in Matlab*, LNCS 3514 (2005), pp. 263–270.
[20] V.I. Arnol'd, *Geometrical Methods in the Theory of Ordinary Differential Equations*, Springer-Verlag, New York, 1983.
[21] H. Broer, and G. Vegter, *Subordinate Šil'nikov bifurcations near some singularities of vector fields having low codimension*, Ergodic Theory Dyn. Sys. 4 (1984), pp. 509–525.
[22] E. Fehlberg, *Classical fifth-, sixth-, seventh-, and eighth order Runge-Kutta formulas with step size control*, Computing 4 (1969), pp. 93–106.
[23] P.J. Prince, and J.R. Dorman, *High order embedded Runge-Kutta formulae*, J. Comp. Appl. Math. 7 (1981), pp. 67–75.
[24] C. Steinmetz, and R. Larter, *The quasiperiodic route to chaos in a model of the peroxidase-oxidase reaction*, J. Chem. Phys. 94 (1991), pp. 1388–1396.
[25] W. Govaerts, and B. Sautois, *Phase response curves, delays and synchronization in Matlab*, LNCS 3992 (2006), pp. 391–398.
[26] W. Govaerts, and B. Sautois, *Computation of the phase response curve: A direct numerical approach*, Neural Comput. 18 (2006), pp. 817–847.
[27] B. Ermentrout, *Simulating, Analyzing, and Animating Dynamical Systems*, Siam Publications, Philadelphia, 2002.
[28] C. Morris, and H. Lecar, *Voltage oscillations in the barnacle giant muscle fiber*, Biophys. J. 35 (1981), pp. 193–213.
[29] Yu.A. Kuznetsov, *Numerical normalization techniques for all codim 2 bifurcations of equilibria in ODEs*, SIAM J. Numer. Anal. 36 (1999), pp. 1104–1124.
[30] R.M. Borisyuk, *Stationary Solutions of a System of Ordinary Differential Equations Depending Upon a Parameter*, Research Computing Centre, USSR Academy of Sciences, Pushchino, 1981. [In Russian]. Fortran Software Series 6.
[31] J. Muñoz-Almaraz, E. Freire, J. Galán, E.J. Doedel, and A. Vanderbauwhede, *Continuation of periodic orbits in conservative and Hamiltonian systems*, Phys. D 181 (2003), pp. 1–38.
[32] A.R. Champneys, and Yu.A. Kuznetsov, *Numerical detection and continuation of codimension-two homoclinic orbits*, Int. J. Bifurcat. Chaos 4 (1994), pp. 785–822.
[33] A.R. Champneys, Yu.A. Kuznetsov, and B. Sandstede, *A numerical toolbox for homoclinic bifurcation analysis*, Int. J. Bifurcat. Chaos 6 (1996), pp. 867–887.
[34] W.-J. Beyn, *Numerical analysis of homoclinic orbits emanating from a Takens-Bogdanov point*, IMA J. Numer. Anal. 14 (1994), pp. 381–410.
[35] W.J. Beyn, A.R. Champneys, E. Doedel, W. Govaerts, Yu.A. Kuznetsov, and B. Sandstede, *Numerical continuation and computation of normal forms*, in *Handbook of Dynamical Systems: Vol 2*, B. Fiedler, ed, Amsterdam, Elsevier, 2002, pp. 149–219.
[36] J.W. Demmel, L. Dieci, and M.J. Friedman, *Computing connecting orbits via an improved algorithm for continuing invariant subspaces*, SIAM J. Sci. Comput. 22 (2001), pp. 81–94.
[37] M. Koper, *Bifurcations of mixed-mode oscillations in a three-variable autonomous Van der Pol-Duffing model with a cross-shaped phase diagram*, Phys. D 80 (1995), pp. 72–94.
[38] J.A. Connor, D. Walter, and R. McKown, *Modifications of the Hodgkin-Huxley axon suggested by experimental results from crustacean axons*, Biophys. J. 18 (1977), pp. 81–102.