A review of numerical continuation and bifurcation analysis software for computational biology problems

Mark

February 21, 2020

ZZZ Currently c.3000 words, 47 refs. Cell systems has a review word limit of 8000 words, 100 refs.

TODO come up with a less verbose title

Perhaps send the preprint to whoever wrote each of the tools, to make sure I haven't missed anything / got anything wrong?

1 TODO Abstract

Abstract

2 TODO Introduction to bifurcation analysis

2.1 TODO : PROOF READ; [intro, remove this header]

Computational biology uses mathematical tools to understand the processes that give rise to living organisms [1]. One area having seen significant success is the application of ideas from nonlinear dynamics and dynamical systems theory to biological systems. Here, processes are described in terms of differential equations, which describe how some aspect of interest evolves in time; an introduction to the field can be found in [2, 3]. The resulting equations can be analysed both to explain observed phenomena, and to predict novel, unseen behaviours. Classic examples of this are the work of Hodgkin and Huxley [4], which laid the foundations for classical neuroscience; the Mackey-Glass equation [5], which studies the effects of time-delayed feedback on respiratory and hematopoietic diseases; and the Lotka-Volterra model [6] of population dynamics.

The behaviour of all these systems depends on the value of various parameters, such as injected current for a neuron model, or population growth rate in ecological models. When a change in parameter values causes a change in system behaviour, a bifurcation is said to have occurred; see [2, 3, 7] for an entry point into the literature. Bifurcations are of great interest in biology [8]. They can contribute to a variety of diseases such as schizophrenia [9], Alzheimer's and Parkinson's disease [10], and epilepsy [11, 12]. Mackey and Milton refer to these abnormal behaviours, arising from unwanted bifurcations, as dynamical diseases [13]; progress can be made into understanding these diseases by considering the systems and the bifurcations that lead to them. Bifurcations can also be used to explain the causes of neuronal spiking and bursting behaviours [14, 15]. The saddle-node bifurcation can explain bistability in metabolic pathways [16], visual perception tasks [17], epigenetic regulation [18], as well as many other biological feedback systems [19].

Clearly therefore, bifurcations are a powerful explanatory tool in systems biology. Nevertheless, the detection and analysis of these bifurcations is often analytically challenging. Consequently, numerous computational algorithms and software packages have been produced to detect and analyse the bifurcations occuring in a system of equations. While some publications include a survey of continuation tools (see [20, 21, 22]), they focus primarily on the history and design approaches of the software. This work aims to complement these studies, by providing a review of the most commonly used bifurcation analysis software, and demonstrating how each package can be applied to problems from mathematical biology. The most commonly used packages are PyDSTool [23], XPPAUTO [24], MatCont [22], and CoCo [25]. These four packages form the core of this work, however other packages also exist, some of which are discussed in section 6.

2.2 TODO : PROOF READ; What is a bifurcation and a dynamical system?

This section aims to provide an intuitive introduction to bifurcation theory; it is aimed at the non-specialist, and hence any rigorous analysis is left for the literature. A more formal introduction to bifurcation theory can be found in [2, 3, 7]; an exposition from a computational biology perspective can be found in [26, 27, 15]. The author particularly recommends [2] for a broad and very readable introduction to the field.

Bifurcation theory considers changes in the dynamics of a system when a parameter is varied. A change is said to have occurred when the before and after dynamics are not equivalent. A topological definition of equivalence is typically adopted – two systems are said to be topologically equivalent if and only if there exists a homeomorphism between their phase portraits. One can loosely imagine this to mean that two systems have equivalent dynamics, only if they can be stretched, rotated, and bent into the one another. While the temporal nature of the dynamics may vary between topologically equivalent systems (eg. slow or rapid convergence to equilibria), their general features, such as stability and existence of limit cycles, equilibria et cetera will remain the same. For example, while the dynamics of a damped fast-swinging pendulum can easily be transformed into that of a damped slow-swinging pendulum, neither can be transformed into a pendulum whose swing amplitude increases in time due to resonance; the fast- and slow-swinging pendula are equivalent, and different to the resonantly excited pendulum.

A system is said to be structurally stable if it is topologically equivalent to any small perturbation within the same class of systems. That is, the inclusion of additional small terms, or the small perturbation of existing terms, does not produce topologically different dynamics within the system. Consider the system of differential equations given by

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

where $x \in \mathbb{R}^n$ is the system state, and $\lambda \in \mathbb{R}^m$ is a vector of system parameters. A bifurcation occurs at parameter value λ^* if the system loses structural stability when $\lambda = \lambda^*$. That is, arbitrarily small variations away from λ^* will produce a system with different dynamics to those at λ^*. A bifurcation diagram shows how invariant sets change as a function of parameter value; a bifurcation set shows the sets in parameter space where a system loses structural stability through bifurcation. **ZZZ QUESTION: It is possible for a system to not be structurally stable, but also not be at a bifurcation point, eg. $x^3 + \lambda$ at (0,0). Does that make this definition wrong? How should I rephrase this section to make it right?**

The codimension of a bifurcation is the number of parameters that must be varied to generically see that bifurcation. This can equivalently be thought of as the number of parameters that are required in a bifurcation diagram to make it structurally stable. If a bifurcation diagram is never structurally stable, the corresponding bifurcation is said to have infinite codimension. Codimension is used to describe the complexity of a bifurcation. Few tools exist to study higher-codimension bifurcations. **ZZZ QUESTION: Codimension higher than what? Do any tools exist to study codim3 bifurcations?**

2.3 TODO : PROOF READ; How does continuation work / do we find bifurcations numerically?

A basic understanding of continuation is necessary for running numerical bifurcation experiments. Continuation considers the computation of implicitly defined manifolds. Consider the system given in equation (1). An equilibrium is given by $f(x, \lambda) = 0$. Under certain assumptions ***WHAT ASSUMPTIONS? SMOOTHNI*** the implicit function theorem guarantees that we can find some manifold $x(\lambda\backslash)$, giving the location of the equilibrium position as a function of the parameter vector λ [28]. Numerical continuation provides a means of computing this manifold, given some starting equilibrium x_0 at parameter value λ_0. In the context of bifurcation analysis, continuation methods are used to track how the solution to a system of equations changes as the parameter vector is varied. The system of equations is given by the vector field f, and a set of test functions whose zeros occur at a bifurcation; additional regularisation constraints, such as phase constraints, are sometimes required to produce a well-posed problem [29]. See [25] ***CHAPTER?*** for a detailed discussion on problem formulation.

Numerical bifurcation analysis is thus a problem of tracking the zeros of a system of equations, test functions, and regularisation constraints, as some of the system parameters change. A numerical investigation of bifurcations generally proceeds by

1. finding equilibria (codimension zero points);

2. tracking those equilibria under single-parameter changes, to reveal codimension 1 bifurcations;

3. tracking these codimension-1 bifurcations in multiple parameters, to find higher codimension singularities.

It is useful to understand bifurcation analysis as being a problem of computing implicitly defined manifolds, as it explains why numerical bifurcation analysis must be approached in this fashion. One must first find points where equilibria bifurcates, by tracking the equilibria as a parameter changes WHICH IS COMPUTING AN IMPLICIT MANIFOLD. This will indicate the locations of codimension-1 bifurcations, which occur at zero-dimensional points in a one-dimensional parameter space. By augmenting the equilibrium problem with test functions, these bifurcations can then be tracked as two parameters are varied WHICH IS COMPUTING AN IMPLICIT MANIFOLD IN THE AUGMENTED SYSTEM. Higher-codimension bifurcations form organising centers, from which manifolds of lower-codimension bifurcations emerge. One therefore proceeds by finding and tracking interesting points in successively higher codimension. **ZZZ TODO fix the bits in caps in the above paragraph**

Continuation is performed using a psuedo-arclength predictor corrector scheme [30]. Here, the next point on the manifold is estimated using its tangent at the previous point, and a corrector scheme is used to refine this estimate. This is designed to replicate the parameterisation of the manifold in terms of arc length from the initial point, allowing for continuation around fold points. While numerical psuedo-arclength continuation is the standard method of bifurction analysis, several variations to this approach have been developed. These include control-based continuation algorithms, for investigating the bifurcation structure of physical and black-box systems [31, 32, 33, 34], and inverse bifurcation algorithms, for discovering parameter configurations that produces some target bifurcation structure. Control-based continuation has recently been applied to an *in silico* gene regulatory network [35]. An introduction to inverse problems is given in [36, 37], with [37] giving a discussion of their applications to systems biology. Inverse methods have been applied variously to designing gene networks that exhibit a specific set of dynamics [38], and producing systems that are robust to parameter noise [39]. An entirely different approach to bifurcation analysis is taken in [40], where a genetic algorithm is used to search for bifurcations in the parameter space of ODE reaction networks.

3 TODO an overview of the available tools, their strengths and limitations, usages, etc.

3.1 TODO : PROOF READ; PyDSTool

TODO HOW MUCH OF THIS IS ACTUALLY NECESSARY INFORMATION? PyDSTool provides a suite of tools for the simulation and analysis of dynamical systems, with a focus on biological applications [23]. It is written primarily in Python3, however legacy C and Fortran code is included for efficient numerical solvers. Being written in Python3, PyDSTool is particularly easy to adapt and extend to new problems. The code is released under the permissive BSD 3-clause license, which allows for modification and redistribution of the source code. PyDSTool supports ordinary differential equations, differential algebraic equations, discrete maps, and hybrid models thereof. Limited support is also available for delay differential equations, however these do not form the focus of PyDSTool; more specialised packages such as pydelay [41], Knut [42], or DDE-BIFTOOL [43] are better suited to such problems.

TODO REPHRASE THIS PARA PyDSTool has no graphical user interface. Instead, modelling and analysis procedures are specified through Python scripts. This has the advantage of allowing for more sophisticated and complex analysis routines than could be achieved with a graphical interface. It makes it easy to rapidly run and rerun analyses, reproduce research, and to change the model and any aspects of its analysis, without having to repeat the entire procedure from scratch; instead, the script can simply be rerun after any desired alterations have been made. Rich data structures are provided to facilitate this. These can be integrated into other work, to extend the capabilities of PyDSTool, and to apply its methods and routines to other problems.

TODO IS THIS TOO MUCH INFORMATION ABOUT HYBRID SYSTEMS? Unlike other software packages, PyDSTool offers advanced support for hybrid models. These can be considered loosely as a set of different regimes of smooth dynamics, and a set of events-based rules to dictate when and how transitions between these regimes should occur; see [44] for a rigorous treatment of hybrid dynamical systems. Hybrid modelling can allow one to express key system behaviours in a significantly simpler way than could otherwise be achieved. A key example of this is the integrate and fire neuron

(see [45] section 1.3). Here, a neuron is modelled as integrating any applied current, and firing a spike when the membrane potential exceeds a threshold. The membrane potential then resets to a resting state. This non-smooth model succeeds in abstracting away the complex dynamics of spike generation, resulting in a simple model that still captures the essence of neuronal behaviours. While hybrid modelling is a powerful tool for biological analysis, only PyDSTool is explicitly designed for building and analysing these models.

TODO GENERALLY OKAY. SHOULD I TALK MORE ABOUT THE PHASE PLANE / COMP N

Models are specified symbolically; symbolic expression routines exist for manipulating derivatives, substitutions, evaluations, and simplifications. Individual models can optionally be combined together, to form hybrid models. These models can then be simulated and analysed. Toolboxes exist for a range of purposes, with examples including parameter fitting and estimation, compartmental modelling of neurons and chemical synapses, and phase plane analysis. Continuation methods are implemented to detect and track bifurcations in parameter space; these are considered in more depth in section 4.

3.2 TODO : PROOF READ; XPPAUTO

XPP (also referred to as XPPAUT, XPPAUTO) is a combined simulation and continuation package [24]. It is one of the oldest dynamical systems tools to still see regular use, and as a result, has seen extensive use for solving and analysing problems across nonlinear dynamics. A large number of tutorials and resources are available because of this. Nevertheless, the age of the software also lends itself to a somewhat *'clunky'* user interface. The program also has a tendency to crash; no scripting interface is available, which means that in the event of a crash, one must restart an entire analysis from scratch.

XPP is capable of handing a wide range of system classes, including ordinary, delay, and stochastic differential equations, boundary value problems, and difference and functional equations. The package is written in C, and source code is released under the GNU GPL v3 license, allowing for modification and redistribution. Nevertheless, the code base does not easily lend itself towards being extended and adapted to novel problems. XPP is used through a graphical interface and models are specified symbolically in text files, meaning no knowlege of coding is required to use the software. Furthermore, XPP provides a graphical interface to most features of AUTO [22, 46], allowing users to run continuation and bifurcation analyses without writing any Fortran code.

XPP has a wide range of features, both within and in addition to simulation and bifurcation analysis. Over a dozen solvers are available, covering forward and backward integration for a range of stiff and non-stiff classes of system. Tools are also provided for phase plane analysis, such as nullcline, vector field, and flow field plotting, and equilibrium location methods. Methods exist to create Poincaré sections and animations directly from XPP.

3.3 TODO : PROOF READ; MatCont

IS THIS TOO OPINION-BASED? MatCont focuses on providing a comprehensive set of simulation and analysis tools, for ordinary differential equations only. The package aims to overcome the shortfalls of previous bifurcation analysis tools (such as detection, continuation, and normal form calculations of codimension-2 bifurcations), and is thus the most fully featured of the tools considered here. The intuitive graphical interface lends itself towards a more gentle learning curve than for other tools. MatCont is freely available under the Creative Commons BY-NC-SA 3.0 license, allowing users to modify and redistribute the software, subject to constraints. Note that MatCont is written for use with MATLAB, and thus requires a MATLAB license; the author was unable to run MatCont in GNU Octave in its provided form.

While being the most powerful tool of those discussed here, MatCont is also extensively and clearly documented, and a large number of tutorials are available. It is available both as a graphical package, and as the command-line version Cl_MatCont. This means that users are not required to write any code to use MatCont; nevertheless, the graphical interface merely acts as an intermediary between the user and Cl_MatCont, so analyses can be carried out just as effectively in a scripting environment, should the user desire. Cl_MatCont also allows one to extend the functionality of the software, by integrating MatCont routines into custom projects.

MatCont has a well-designed memory management system, allowing curves and points of interest to be saved automatically. As a result, users are not forced to re-run analyses from scratch each time a system is studied, even when working from the graphical interface.

Models are provided symbolically to MatCont. The software is able to compute derivatives symbolically, allowing for faster code execution, and improved precision. It supports additional features such as Poincaré maps and phase repsonse curves, and is the only software to support normal form analysis of limit cycle bifurcations, using the methods developed in [47]. Users have access to all MATLAB solvers, in addition to two additional Runge-Kutta solvers for stiff systems.

3.4 TODO CoCo

ZZZ

- **Background to the software - who wrote it, what for, when, why?**
- **What systems can it analyse?**
- **What does it try to do / solve? What niche was it created to fill?**
- **What problems / users is it aimed at?**
- **What nice touches does it have to make it worth using? Toolboxes? Scripting? Exports? Simulation? Phase planes?**
- **Only describe the tools here, don't compare them to each other!**
- **CoCo is only an analysis environment, though it's used through matlab, which simulations could be run in**
- **requires user to code the problem in matlab**
- **functions, variables, etc. are declared in Matlab syntax**

4 TODO a comparison of those tools and their functionality

ZZZ Notes to work into the text somewhere:

- **PyDSTool and XPP integrators are significantly faster than CoCo / MatCont's matlab integrators.**
- **PyDSTool can support arbitrarily large models, which XPP can't**
- **MatCont and PyDSTool's bifurcation analysis is closer-integrated with its simulation / core tools than for XPP**
- **CoCo, PyDSTool and $CL_{MatCont}$ can be scripted, and integrated into other programs; XPP cannot easily, however some attemts at interfaces exist (see website).**
- **XPP and MatCont have GUIs; PyDSTool and CoCo do not**

4.0.1 Types of system each software can handle

System	MatCont	CoCo	XPP	PyDSTool
ODE	y	?	y	y
PDE (discretized)	n	?	y	n
DDE	n	?	y	limited
SDE	n	?	y	limited
DAE	n	?	y	y
BVP	n	?	y	n
Maps	MatContM	?	y	y
Hybrid	n	?	basic (apparently)	y
Integral	n	?	y	n
Functional	n	?	y	n

ZZZ QUESTION I'm yet to look at what CoCo is and isn't capable of analysing. Ask Ludovic for some pointers?

QUESTION While XPP is capable of simulating all the noted systems, I don't know if that is literally just XPP simulating them, or also that AUTO is able to run continuations with them. Ask supervisors for advice.

4.0.2 Types of point each software can study

- D: software can detect this invariant set
- C: software can continue this invariant set

Point label	Point type	Codim	MATCONT	CoCo	XPP	PyDSTool
EP	Equilibrium	0	D,C		D,C	D,C
LC	Limit cycle	0	D,C		D,C	D,C
LP	Limit point	1	D,C		D,C	D,C
H	Hopf	1	D,C		D,C	D,C
LPC	Limit point of cycles	1	D,C		?-?	D
NS	Neimark-Sacker	1	D,C		???	D,**
Torus bif	?????????????		??????		D,???	???????
PD	Period doubling	1	D		D,C	D,**
BP *	Branch point		D,C		D,?	D,?
CP	Cusp	2	D		?-?	D
BT	Bogdanov Takens	2	D		?-?	D
ZH	Zero-Hopf	2	D		?-?	D
HH	Double Hopf	2	D		?-?	D
GH	Generalised Hopf	2	D		?-?	D
BPC *	Branch point of cycles		D		?-?	?
CPC	Cusp point of cycles	2	D		?-?	-
CH	Chenciner	2	D		?-?	-
LPNS	Fold-Neimark-Sacker	2	D		?-?	-
PDNS	Flip-Neimark-Sacker	2	D		?-?	-
LPPD	Fold-Flip	2	D		?-?	-
NSNS	Double Niemark-Sacker	2	D		?-?	-
GPD	Generalised flip	2	D		?-?	-

ZZZ

- **QUESTION: Is torus bifurcation the same as Neimark-Sacker, or are NS a map bifurcation and torus the flows equivalent?**
- *** QUESTION: Are branch points just 'there's a bifurcation here but we don't know what type specifically'? In that case, any bifurcation that occurs, but isn't one of the labelled ones, would still be detected as a BP.**
- **Also see the MATCONT 'objects related to homoclinics to equilibria' table, and resonances, for additional *stuff* it can detect / continue**
- **** QUESTION: PyDSTool seems to have methods to continue these for fixed points of maps; does that mean they're a maps-only type of curve? Note that it lacks documentation and tests/examples about these methods, so maybe they're not implemented?**

5 TODO examples of using those tools with the Hindmarsh-Rose model

6 TODO allude to the existence of non-ODE packages

ZZZ How valuable is this section? Should I delete it entirely? If not, are there tools for stochastics? Any additional tools for PDEs / nonsmooth systems?

This work does not intend to provide a review of all dynamical systems software. Nevertheless, common continuation tools for other classes of systems are included here for completeness.

6.1 Other ODE tools

ZZZ Justify why I haven't studied these here. Good refs in that textbook about the history of these softwares.

- **DSTool**
- **CONTENT**
- **AUTO**
- **...**

6.2 Delay differential equations

ZZZ Brief note about why DDEs are relevant to biology.

- **Engelborghs, Koen, et al. "Numerical bifurcation analysis of delay differential equations arising from physiological modeling." Journal of mathematical biology 42.4 (2001): 361-385.**
- **Luzyanina, Tatyana, Dirk Roose, and Gennady Bocharov. "Numerical bifurcation analysis of immunological models with time delays." Journal of computational and applied mathematics 184.1 (2005): 165-176.**

DDE BIFTOOL [43] is a MATLAB package for analysis of systems with fixed delays. It provides stability analysis and tracking of equilibrium and limit cycle solutions, and is capable of tracking bifurcations. The interface is script-based. Knut [42] provides a graphical package for both analysing and simulating DDEs. It supports stability analysis, orbit continuation, and bifurcation detection in one parameter, and has methods for the two-parameter continuation of some bifurcatons. Unlike DDE BIFTOOL, it requires no programming knowlege to use; being written in C++, it is also faster. Pydelay [41] provides an easy-to-use package for simulating DDEs in Python3, featuring automatic low-level code generation for efficient solving. It lacks any analysis tools.

6.3 Partial differential equations

ZZZ Brief note about where PDEs arise in biology

- **PDECONT**

6.4 Non-smooth systems

ZZZ Brief note about where non-smooth dynamics arise in biology

- **SLIDECONT**
- **TC HAT**

6.5 Large-scale biological systems

ZZZ Find references comparing all these software. Presumably there's some.

- **Neuron**
- **NEST**
- **VCell**
- **Bio-SPICE**
- **Brian**
- **Chaste**
- **SloppyCell**

7 TODO concluding remarks

References

[1] Daniel A Beard, James B Bassingthwaighte, and Andrew S Greene. Computational modeling of physiological systems, 2005.

[2] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[3] John Guckenheimer and Philip Holmes. *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*, volume 42. Springer Science & Business Media, 2013.

[4] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[5] Michael C Mackey and Leon Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

[6] Vito Volterra. Variations and fluctuations of the number of individuals in animal species living together. *ICES Journal of Marine Science*, 3(1):3–51, 1928.

[7] Yuri A Kuznetsov. *Elements of applied bifurcation theory*, volume 112. Springer Science & Business Media, 2013.

[8] Michael R Guevara. Bifurcations involving fixed points and limit cycles in biological systems. In *Nonlinear Dynamics in Physiology and Medicine*, pages 41–85. Springer, 2003.

[9] Uwe An der Heiden. Schizophrenia as a dynamical disease. *Pharmacopsychiatry*, 39(S 1):36–42, 2006.

[10] IH Mustafa, G Ibrahim, A Elkamel, SSEH Elnashaie, and P Chen. Non-linear feedback modeling and bifurcation of the acetylcholine neurocycle and its relation to alzheimer's and parkinson's diseases. *Chemical Engineering Science*, 64(1):69–90, 2009.

[11] Fernando Lopes Da Silva, Wouter Blanes, Stiliyan N Kalitzin, Jaime Parra, Piotr Suffczynski, and Demetrios N Velis. Epilepsies as dynamical diseases of brain systems: basic models of the transition between normal and epileptic activity. *Epilepsia*, 44:72–83, 2003.

[12] John G Milton. *Epilepsy: Multistability in a dynamic disease*. Cambridge, UK: Cambridge Univ. Press, 2000.

[13] Michael C Mackey and John G Milton. Dynamical diseases. *Neuro—ophthalmology*, 21:24, 1987.

[14] Eugene M Izhikevich. Neural excitability, spiking and bursting. *International journal of bifurcation and chaos*, 10(06):1171–1266, 2000.

[15] Eugene M Izhikevich. *Dynamical systems in neuroscience*. MIT press, 2007.

[16] Orlando Díaz-Hernández and Moisés Santillán. Bistable behavior of the lac operon in e. coli when induced with a mixture of lactose and tmg. *Frontiers in physiology*, 1:158, 2010.

[17] Dante R Chialvo and A Vania Apkarian. Modulated noisy biological dynamics: three examples. *Journal of Statistical Physics*, 70(1-2):375–391, 1993.

[18] Daniel Jost. Bifurcation in epigenetics: implications in development, proliferation, and diseases. *Physical Review E*, 89(1):010701, 2014.

[19] David Angeli, James E Ferrell, and Eduardo D Sontag. Detection of multistability, bifurcations, and hysteresis in a large class of biological positive-feedback systems. *Proceedings of the National Academy of Sciences*, 101(7):1822–1827, 2004.

[20] Hil Meijer, Fabio Dercole, and Bart E Oldeman. Numerical bifurcation analysis., 2009.

[21] Willy Govaerts and Yuri A Kuznetsov. Interactive continuation tools. In *Numerical continuation methods for dynamical systems*, pages 51–75. Springer, 2007.

[22] Annick Dhooge, Willy Govaerts, Yu A Kuznetsov, Hil Gaétan Ellart Meijer, and Bart Sautois. New features of the software matcont for bifurcation analysis of dynamical systems. *Mathematical and Computer Modelling of Dynamical Systems*, 14(2):147–175, 2008.

[23] Robert Clewley. Hybrid models and biological model reduction with pydstool. *PLoS computational biology*, 8(8), 2012.

[24] Bard Ermentrout. *Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students*, volume 14. Siam, 2002.

[25] Harry Dankowicz and Frank Schilder. *Recipes for continuation*, volume 11. SIAM, 2013.

[26] Anne Beuter, Leon Glass, Michael C Mackey, and Michele S Titcombe. *Nonlinear dynamics in physiology and medicine*. 2003.

[27] Frank C Hoppensteadt and Eugene M Izhikevich. *Weakly connected neural networks*, volume 126. Springer Science & Business Media, 2012.

[28] Eugene L Allgower and Kurt Georg. *Introduction to numerical continuation methods*, volume 45. SIAM, 2003.

[29] Eusebius Doedel, Herbert B Keller, and Jean Pierre Kernevez. Numerical analysis and control of bifurcation problems (i): Bifurcation in finite dimensions. *International journal of bifurcation and chaos*, 1(03):493–520, 1991.

[30] Herbert B Keller and KELLER HB. Numerical solution of bifurcation and nonlinear eigenvalue problems. 1977.

[31] Kestutis Pyragas. Continuous control of chaos by self-controlling feedback. *Physics letters A*, 170(6):421–428, 1992.

[32] Kestutis Pyragas. Control of chaos via an unstable delayed feedback controller. *Physical Review Letters*, 86(11):2265, 2001.

[33] David AW Barton and Jan Sieber. Systematic experimental exploration of bifurcations with noninvasive control. *Physical Review E*, 87(5):052916, 2013.

[34] Jan Sieber and Bernd Krauskopf. Control based bifurcation analysis for experiments. *Nonlinear Dynamics*, 51(3):365–377, 2008.

[35] Brandon Gomes, Irene de Cesare, Agostino Guarino, Mario di Bernardo, Ludovic Renson, and Lucia Marucci. Exploring the dynamics of nonlinear biochemical systems using control-based continuation. *bioRxiv*, page 695866, 2019.

[36] Heinz W Engl and Philipp Kügler. Nonlinear inverse problems: theoretical aspects and some industrial applications. In *Multidisciplinary methods for analysis optimization and control of complex systems*, pages 3–47. Springer, 2005.

[37] Heinz W Engl, Christoph Flamm, Philipp Kügler, James Lu, Stefan Müller, and Peter Schuster. Inverse problems in systems biology. *Inverse Problems*, 25(12):123014, 2009.

[38] James Lu, Heinz W Engl, and Peter Schuster. Inverse bifurcation analysis: application to simple gene systems. *Algorithms for molecular biology*, 1(1):11, 2006.

[39] Hiroyuki Kitajima and Tetsuya Yoshinaga. A method for constructing a robust system against unexpected parameter variation. In *Analysis and Control of Complex Dynamical Systems*, pages 41–48. Springer, 2015.

[40] Vijay Chickarmane, Sri R Paladugu, Frank Bergmann, and Herbert M Sauro. Bifurcation discovery tool. *Bioinformatics*, 21(18):3688–3690, 2005.

[41] V. Flunkert and E. Schöll. pydelay – a python tool for solving delay differential equations. `arXiv:0911.1633 [nlin.CD]`, 2009.

[42] R Szalai. Knut: a continuation and bifurcation software for delay-differential equations (version 8), department of engineering mathematics, university of bristol (2013).

[43] Koen Engelborghs, Tatyana Luzyanina, and Dirk Roose. Numerical bifurcation analysis of delay differential equations using dde-biftool. *ACM Transactions on Mathematical Software (TOMS)*, 28(1):1–21, 2002.

[44] Slobodan N Simic, Karl Henrik Johansson, John Lygeros, and Shankar Sastry. Towards a geometric theory of hybrid systems. *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, 12(5-6):649–687, 2005.

[45] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[46] Eusebius J Doedel. Auto: A program for the automatic bifurcation analysis of autonomous systems. *Congr. Numer*, 30(265-284):25–93, 1981.

[47] Yu A Kuznetsov, Willy Govaerts, Eusebius J Doedel, and Annick Dhooge. Numerical periodic normalization for codim 1 bifurcations of limit cycles. *SIAM journal on numerical analysis*, 43(4):1407–1435, 2005.