DIANA — An object-oriented tool for nonlinear analysis of chemical processes

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

von M.Sc. Mykhaylo Krasnyk

geb. am 22.01.1982 in Donezk, Ukraine

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik der Otto-von-Guericke-Universität Magdeburg

Gutachter:	Prof. Dr.-Ing. Achim Kienle
	Prof. Dr.-Ing. Martin Mönnigmann
	Dr.-Ing. habil. Michael Mangold

Promotionskolloquium am 2. September 2008

Forschungsberichte aus dem Max-Planck-Institut
für Dynamik komplexer technischer Systeme

Band 23

Mykhaylo Krasnyk

DIANA — An object-oriented tool for nonlinear analysis of chemical processes

Shaker Verlag
Aachen 2008

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Zugl.: Magdeburg, Univ., Diss., 2008

ISBN 978-3-8322-7692-8
ISSN 1439-4804

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen
Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9
Internet: www.shaker.de • e-mail: info@shaker.de

Preface

This thesis is the result of four years of work in the field of numerical nonlinear analysis at the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg and would not have been possible without the help of many.

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Achim Kienle. Without his advice and support this thesis would never had become reality. Further, I would like to thank my co-supervisor Dr.-Ing. habil. Michael Mangold for his great help and support during my work at the Max Planck Institute. Also I would like to thank Martin Ginkel for helpful discussions on diverse topics in modeling systems and software engineering, especially, Lisp programming.

I am grateful to Prof. Dietrich Flockerzi for his pleasant lectures in nonlinear dynamics. I also wish to thank Prof. V. A. Svjatnyj from the Donetsk National Technical University for the Pro3 scholarship offer that gave me the possibility to visit Germany during the master study.

I wish to thank all my colleagues who helped to create the unique atmosphere at the group which allows one to pursue his research successfully. In particular I wish to thank Carolyn Mangold for all her help she did for me.

Last but certainly not least, I thank my parents for their constant love and care.

Magdeburg, September 2008 Michael Krasnyk

To my parents

Моим родителям

Contents

1 Introduction **1**

1.1 Motivation of the thesis 1

1.2 A survey of existing tools 3

1.3 The numerical analysis tool Diana 4

1.4 Outline of the thesis 6

2 Parameter Continuation Methods and Bifurcations of Equilibria **9**

2.1 Model description 10

2.2 Parameter continuation methods 13

2.2.1 Predictors 15

2.2.2 Correctors 16

2.2.3 The step size control 18

2.2.4 The continuation algorithm 19

2.3 Steady-state continuation and the linearized stability analysis . . . 20

2.4 The Lyapunov-Schmidt reduction 23

2.5 The singularity recognition and unfolding 26

2.6 Symbolic differentiation 30

2.7 The numerical singularity computation 32

2.8 The Hopf bifurcation 35

2.9 Continuation of the Hopf bifurcation point 38

2.10 Singularities of the Hopf bifurcation point 39

2.11 Conclusion 41

3 Periodic Orbits Continuation and the Recursive Projections Method **45**

3.1 Periodic solutions 46

3.2 Stability and bifurcations of periodic orbits 50

3.3 Continuation of periodic solutions 53

3.4 The recursive projection method 55

3.4.1 Mathematical background 56

3.4.2 Numerical algorithm 59
3.5 Application of the recursive projection method 62
3.5.1 Linear algebraic problem 62
3.5.2 Calculation of steady-state solutions 64
3.5.3 Calculation of periodic solutions 64
3.5.4 Show case example 65
3.6 Conclusion 68

4 Case Studies 71
4.1 Nonlinear analysis of a CSTR 72
4.1.1 Dynamic simulation of a CSTR 73
4.1.2 Singularity Analysis of a CSTR 75
4.2 Singularity analysis of a Molten Carbonate Fuel Cell 82
4.2.1 MCFC model 83
4.2.2 Cell operation at constant voltage 84
4.2.3 Cell operation at constant current 90
4.3 Periodic solutions in a MSMPR Crystallizer 93

5 Conclusion and Future Directions 101
5.1 Conclusion 101
5.2 Suggestions for future work 103

A Software Architecture 107
A.1 Class diagrams of Diana 107
A.2 Diana simulation models 113
A.3 Diana dynamic solvers 115
A.4 Diana continuation solvers 117
A.4.1 Steady-state continuation solver 118
A.4.2 Singularity analyzer solver 119
A.4.3 Hopf point continuation solver 120

Bibliography 121

Index 129

Abstract

Chemical production processes often show a strongly nonlinear behaviour. Understanding, controlling, or even exploiting this behaviour can improve the productivity and safety of a process. Numerical bifurcation analysis has become in the last two decades a well-established mathematical tool for the nonlinear analysis of process models. However, traditional bifurcation packages, *e.g.,* AUTO (D^+02), are suitable only for low order systems of ODEs, whereas realistic chemical models comprise hundreds or thousands of differential equations. Continuation methods for huge order systems have been published, for example, LOCA (S^+02), DIVA (MKMG00), but are restricted to the continuation of simple singularities, like limit points and Hopf bifurcations.

This dissertation focuses on the bifurcation analysis of dynamical systems described by differential-algebraic equation systems with applications to chemical engineering models. The objective of this thesis is to develop a nonlinear analysis software environment Diana with a user-friendly interface for dealing with complex nonlinear phenomena in engineering sciences. The key aspects of this environment are object-oriented models, an efficient numerical kernel, and the use of the scripting language Python as a powerful-command line interface. Models are created with the object-oriented and equation based modeling tool ProMoT (TZGG00), that allows to build object-oriented models, described by differential-algebraic equations. Different nonlinear algebraic and differential-algebraic solvers are implemented in Diana, based on existing third party open-source code and can be applied by to the model depending on the special properties of the model.

The nonlinear solvers suite is presented is presented with solvers for the continuation of steady-state points of continuous dynamical systems, limit points and singularities of higher codimension, Hopf points and periodic solutions. The solvers can treat high-dimensional models with help of reduction methods, like, the Lyapunov-Schmidt reduction (GS85) for the singularity analysis, or the recursive-projection method (SK93) for the continuation of periodic solutions. The required higher order derivatives are obtained analytically via an interface to the computer

algebra system Maxima (FdSMY04), therefore it is possible to find bifurcation varieties with high accuracy.

The use of the tool will be illustrated by the analysis of three chemical engineering process models: a continuous flow stirred tank reactor model, a spatially distributed model of a high temperature fuel cell, and a crystallizer in continuous mode of operation with fines dissolution and classified product removal.

Zusammenfassung

Chemische Produktionsprozesse zeigen oft ein stark nichtlineares Verhalten. Das Verständnis dieses Verhaltens, seine Beherrschung oder gar seine technische Nutzung können die Produktivität und die Sicherheit eines Prozesses verbessern. Die numerische Bifurkationsanalyse ist in den letzten zwei Jahrzehnten zu einem etablierten mathematischen Werkzeug für die nichtlineare Analyse von Prozessmodellen geworden. Jedoch sind traditionelle Bifurkations-Programmpakete, z. B. AUTO nur für Systeme von gewöhnlichen Differentialgleichungen niedriger Ordnung geeignet, wohingegen realistische verfahrenstechnische Modelle aus Hunderten oder Tausenden von Differentialgleichungen bestehen. Es wurden auch Fortsetzungsverfahren für Systeme sehr hoher Ordnung veröffentlicht, z.B. LOCA oder DIVA; diese beschränken sich aber auf die Fortsetzung einfacher Singularitäten wie Umkehrpunkten oder Hopfbifurkationspunkten.

Diese Dissertation konzentriert sich auf die Bifurkationsanalyse dynamischer Systeme, die durch Differential-Algebra-Systeme beschrieben werden. Als Anwendungen werden verfahrenstechnische Modelle betrachtet. Das Ziel der Arbeit ist es, eine Software-Umgebung DIANA für die nichtlineare Analyse zu entwickeln, die eine benutzerfreundliche Bedienoberfläche besitzt und die sich für die Untersuchung komplexer nichtlinearer Phänomene in Ingenieuranwendungen eignet. Wesentliche Eigenschaften dieser Software-Umgebung sind objektorientierte Modelle, ein effizienter numerischer Kern und der Einsatz der Skriptsprache Python als ein leistungsstarkes Kommandozeilen-Interface. Die Modelle werden mit Hilfe des objektorientierten und gleichungsbasierten Modellierungswerkzeug ProMoT erzeugt, das es erlaubt, objektorientierte Modelle aufzubauen, die aus Differential-Algebra-Systemen bestehen. Verschiedene nichtlineare algebraische und differential-algebraische Gleichungslöser werden in DIANA implementiert, die auf freier Software von dritter Seite basieren und die in Abhängigkeit der speziellen Eigenschaften eines Modells eingesetzt werden können.

Ein Paket nichtlinearer Lösungsalgorithmen wird präsentiert, das Methoden zur Fortsetzung stationärer Lösungen kontinuierlicher dynamischer Systeme enthält, sowie Methoden zur Fortsetzung von Umkehrpunkten und Singularitäten höherer Kodimension und zur Fortsetzung von Hopfbifurkationspunkten und periodischen Lösungen. Die Löser können hochdimensionale Probleme behandeln, wobei als Reduktionsmethoden die Lyapunov-Schmidt-Reduktion für die Singularitätsanalyse und die Rekursive Projektionsmethode für die Fortsetzung periodischer Lösungen zum Einsatz kommen. Die erforderlichen Ableitungen höherer Ordnung werden analytisch über eine Schnittstelle zum Computer-Algebra-System Maxima bestimmt; daher ist es möglich, Bifurkationsvarietäten mit hoher Genauigkeit zu ermitteln.

Die Verwendung des Software-Werkzeuges wird anhand der Analyse dreier chemischer Prozessmodelle illustriert: einem kontinuierlichen Rührkesselreaktor, einem örtlich verteilten Modell einer Hochtemperaturbrennstoffzelle sowie einem kontinuierlich betriebenen Kristallisator mit Feinkornauflösung und klassifizierendem Produktabzug.

List of Abbreviations

BVP	–	**B**oundary **V**alue **P**roblem
CAPE	–	**C**omputer **A**ided **P**rocess **E**ngineering
CSTR	–	**C**ontinuous **S**tirred-**T**ank **R**eactor
DAE	–	**D**ifferential-**A**lgebraic **E**quations system
ESO	–	**E**quation **S**et **O**bject
IDA	–	**I**nitial value problem solver for **DA**E systems
IVP	–	**I**nitial **V**alue **P**roblem
MCFC	–	**M**olten **C**arbonate **F**uel **C**ell
MDL	–	**M**odel **D**efinition **L**anguage
MSMPR	–	**M**ixed-**S**olution-**M**ixed-**P**roduct **R**emoval crystallizer
NLD	–	**N**on**l**inear **D**ynamic
ODE	–	**O**rdinary **D**ifferential **E**quation
OOP	–	**O**bject-**O**riented **P**rogramming
RPM	–	**R**ecursive **P**rojection **M**ethod
SOFC	–	**S**olid **O**xide **F**uel **C**ell
UML	–	**U**nified **M**odeling **L**anguage
DIVA	–	**d**ynamischer **S**imulator für **v**erfahrenstechnische **A**nlagen
ProMoT	–	**Proc**ess **Mo**deling **T**ool
DIANA	–	**D**ynamic s**i**mulation **a**nd **N**onlinear **A**nalysis

List of Symbols

This list gives the usual meaning of the symbols used throughout the text. In some cases, another meaning which is not included in this list may be used. Also symbols from Section 4 are not included in the following table.

$d^k f$	k-order derivative of f, 2.4
f	DAE residual in implicit form, 2.1
f_x	derivative of f with respect to x, 2.1
g	reduced scalar function, 2.4
$\ker A$	kernel of a matrix A, 2.4
$n(y)$	parametrization equation, 2.2.2
p	solution vector in an “unstable” subspace in the RPM, 3.4.1
$\bar{p}$	coordinates of p vector in V_p basis, 3.4.1
q	solution vector in a “stable” subspace in the RPM, 3.4.1
$\bar{q}$	coordinates of q vector in V_q basis, 3.4.1
r	residual function, 3.4.1
$\operatorname{range} A$	range of a matrix A, 2.4
s	phase condition, 3.1
t	time or independent variable, 2.1
v_0, v_0^*	right and left eigenvectors that correspond to a critical eigenvalue, 2.4
x	state variables vector, 2.1
$\dot{x}$	derivative with respect to an independent variable, 2.1
x_0	initial variables vector, 2.1
y	extended variables vector, 2.2
$\bar{y}$	tangent vector at point y, 2.2.1
$\vec{y}$	unit tangent vector at point y, 2.2.1
$\tilde{y}$	predicted value of y, 2.2
z	scalar variable, 2.4

E	projection onto range A, 2.4
G	nonlinear constraint, 3.4.1
M	monodromy matrix, 3.1
P	projection onto an "unstable" subspace in the RPM, 3.4.1
Q	projection onto a "stable" subspace in the RPM, 3.4.1
T	period, 3.1
V_p	basis of an "unstable" subspace in the RPM, 3.4.1
V_q	basis of a "stable" subspace in the RPM, 3.4.1
X	state space, 2.1
$\alpha, \beta, \gamma \in \nu$	scalar parameters, 2.5
α_0	real part of a critical eigenvalue, 2.8
β, γ	singular values, 2.7
δ	vector update, 2.2.2
ϵ	convergence thresholds, 2.2.2
ϑ	characteristic exponent, 3.1
$\kappa \in \nu$	model parameter, 2.8
λ	scalar parameter, 2.2
μ	characteristic multipliers, 3.1
ν	parameters vector, 2.1
ξ	scaling factor, 2.2.3
ϱ	the RPM threshold value, 3.4.1
σ	step size, 2.2.3
τ	time shift, 3.1
ϕ	reduced function, 2.4
φ	solution of a DAE system, 3.1
φ^t	evaluation operator, 2.1
ω_0	imaginary part of a critical eigenvalue, 2.8
ξ	scaling factor, 2.2.3
ζ	curve length parameter, 2.2
Π	Poincaré map, 3.2
Λ_i	generalized eigenvalues, 2.3
Ω	Poincaré or transversal section, 3.2

List of Figures

1.1 ProMoT/Diana usage diagram 5
1.2 Dynamic simulation results in Diana 6
1.3 Parameter continuation results in Diana 7

2.1 Illustration of predictors 15
2.2 Parametrization types 17
2.3 Local bifurcations of an equilibrium 22
2.4 Numerical error of derivatives calculation 30
2.5 Flow chart for the the recognition problem 34
2.6 Degenerate Hopf Bifurcations 39

3.1 A Poincaré section Ω and dynamic system trajectories 51
3.2 Eigenvalues of the test matrix $F_x := (I - A)$ 63
3.3 Residual norms for Picard iterations 63
3.4 Continuation of steady states and periodic solutions 66
3.5 Newton-Picard stabilization for the steady state continuation . . . 67
3.6 Newton-Picard stabilization for the periodic orbit continuation . . 68

4.1 Dynamical simulation of the CSTR 75
4.2 The one-parameter continuation of the steady state 76
4.3 The two-parameter continuation of the Hopf point 77
4.4 The limit point continuation 77
4.5 The hysteresis point continuation 79
4.6 Singularity varieties of the CSTR model 81
4.7 The one-parameter continuation of steady states 81
4.8 Temperature dependent electrical conductivity 82
4.9 The high temperature fuel cell scheme 83
4.10 Steady solutions of an infinite length MCFC system 86
4.11 Saddle node bifurcations of an infinite length MCFC system 87
4.12 Steady solutions of an finite length MCFC system 88

4.13 Spatial temperature profiles of the solutions I – V in Figure 4.12 . 89
4.14 The pitchfork variety of the MCFC 90
4.15 Singularity varieties of the MCFC 91
4.16 The steady-state temperature dependence 92
4.17 The steady-state profiles of the MCFC 92
4.18 The continuous crystallizer scheme 94
4.19 The Hopf points continuation 95
4.20 The periodic orbits continuation 96
4.21 The Floquet multipliers of the periodic orbits 97
4.22 The dynamic simulation of the model 99

A.1 The package diagram of the implemented CAPE-OPEN interfaces 108
A.2 Diana models class diagram 110
A.3 The IDA solver class diagram 111
A.4 The Continuation package class diagram 112

List of Tables

2.1 Normal forms for singularities of codim $g \leqslant 3$ 29

3.1 The initial values and parameters for the test examples I – IV . . . 67
3.2 Results of the test examples I – IV 68

4.1 Parameter values used in the simulations of MCFC 85

List of Python Scripts

4.1 The dynamic simulation of the CSTR in Diana 73
4.2 The steady-state continuation of the CSTR model 76
4.3 The limit point continuation of the CSTR model 78
4.4 The hysteresis point location 79

Chapter 1

Introduction

The goal of computing is insight, not numbers,
and little insight can be derived from a
computation whose validity is not known.

Richard Hamming

1.1 Motivation of the thesis

This thesis will be concerned with the development of a software tool for the qualitative analysis of nonlinear differential-algebraic equations that appear in chemical engineering and related sciences. There are no general methods for solving these equations and although great ingenuity has been deployed in the treatment of many types of nonlinear problems, most nonlinear equations remain unresolved. Such nonlinear problems are essentially irreducible to integral form and the necessary numerical solutions of these types of problems exhibit features that have no counterparts in integrable nonlinear equations.

The qualitative analysis of nonlinear differential equation leads to the theory of bifurcations which is not particularly new. The modern era of dynamical systems theory began in 1890 with the work on celestial mechanics of the French mathematician Henri Poincaré who was trying to solve the three-body problem (BG94). His concern was the analysis of the earth-moon-sun system under mutual gravitational attraction. The equations for this system were well known at the time and are relatively simple to write down. Their solutions, Poincaré discovered, were highly sensitive to changes in the initial conditions. The methods developed therein laid the basis for the local and global analysis of nonlinear differential equations. Further development of bifurcation theoretical methods was made by many prominent mathematicians, like George Birkhoff, Aleksandr Andronov, Heinz Hopf, Andrey Kolmogorov, Vladimir Arnold, Jürgen Moser, Stephen Smale and others.

In the last decades considerable progress has been made with nonlinearities. On the one hand, analytical methods have been developed that can extract important information from nonlinear equations without actually solving them. On the other hand, the use of computers has led to important progress in understanding the nature of the solutions of equations that cannot be handled by any analytical method. The use of these methods is now spreading into the more engineering or related sciences. These methods are employed for the detection, identification, and quantification of structural nonlinearities of process engineering models. With methods of nonlinear analysis it is possible to predict and distinguish different behaviors of models. A wide variety of tasks concerning multiple solutions can be reduced to studying the dependence of the solutions of a single scalar equation with respect to the parameters. This technique — known as the Lyapunov-Schmidt reduction (GS85) — allows to find the so-called organizing centers of the model. The organizing center is associated with a distinguished set of values for the parameters such that all possible different qualitative behaviors occur for parameter values in a small neighborhood of the distinguished values. Such kind of points exhibit the most singular behaviour and pseudo-global results may be often obtained by the application of local analysis near the organizing center. Detection and analysis of the organizing center may give insight of the model behavior, stability domains, and the model parameter dependence. In application, for example, such analysis can be used for the computing of stability boundaries in a controller or for the recently presented optimization-based constructive nonlinear dynamics (GMM05). Another topic of the nonlinear analysis is the location of periodic solutions, determination of the parameter dependence and stability of solutions. Such results are important in applications, for example, in crystallization processes, because the oscillations should be avoided for a better product quality. Conversely in special cases the periodic oscillations may be useful for industrial applications (RMK$^+$06; SRP$^+$98).

With evolution of digital computers diverse nonlinear analysis software has been created that are based on nonlinear analysis methods. A representative, but not full, list of existing nonlinear analysis software is given in the next section.

1.2 A survey of existing tools

One of the firsts software tool that can be mentioned is the FORTRAN subprogram DERPAR (Kub76) by Milan Kubíček. The subprogram is intended for calculating the dependence of the solution of a nonlinear ODE system on one parameter. Another FORTRAN package is BIFPACK (Sey93) by Rüdiger Seydel that can handle nonlinear algebraic equations, boundary-value problems of ODE, and autonomous differential equations. The widely used package in bifurcation analysis is AUTO (D^+02) by Eusebius J. Doedel and others. The software package contains algorithms for continuation and bifurcation problems in low-order ordinary differential equations. Another widespread family of nonlinear software is LOCBIF (KKLN93) by Alexander I. Khibnik and others, CONTENT (KL97) by Yuri A. Kuznetsov and Victor V. Levitin, MATCONT (DGK^+06) by Willy Govaerts, Yuri A. Kuznetsov and others. These packages are intended for the parameter continuation of equilibrium and periodic solutions of ordinary differential equations, detection and continuation of bifurcation points, periodic and homoclinic solutions. The packages also implement methods for the analysis of discrete nonlinear maps and their bifurcations. The tool LOCA (S^+02) by A. G. Salinger that is intended to perform bifurcation analysis of large-scale CFD applications. The tool PDECONT (Lus97) by Kurt Lust allows to compute periodic solution of large systems, like discretized PDEs, and to continuate such solutions with respect to a control parameter. The package is written in C language and uses the recursive projection method that exploits the property that systems under study usually have a low-dimensional attractor. This assumption leads to quite efficient numerical code, implemented in PDECONT, that combines cheap iterative methods for a higher-order solution subspace and Newton's method with a direct solver for a lower-order subspace for computing and analyzing periodic solutions. Also should be mentioned the comprehensive chemical engineering software tool Diva (MKMG00). Diva is purposed for both stationary and dynamic simulations of chemical engineering processes that can be described by higher-order differential-algebraic systems. Last not least there are simulation, modeling and analysis packages for dynamical systems, like XPPAUT (Erm02) by Bard Ermentrout and PyDSTool (PyD) that is being developed at the Cornell University.

Existing tools suffer from a number of limitations with respect to to the classes of models that can be analyzed or with respect to the intended users. The majority of existing software packages focuses on ODE systems, while chemical process models are differential-algebraic. Furthermore, the successful application of most available NLD tools requires a deep insight in bifurcation theory from the user. Easy to use software tools for nonlinear analysis of chemical processes are desirable but hardly available. As exception is DIVA, which offer methods for one-parameter continuation of steady-states and periodic solutions as well as for the two parameter continuation of limit points and Hopf bifurcation points. However, DIVA suffer from an outdated software architecture that makes extension of the existing tools very difficult to impossible. For example, DIVA gives no possibility to compute singularities of higher co-dimension and the periodic continuation is only efficient for low-order systems.

A novel tool should have the following properties:

- modularization, extensibility and object-oriented architecture
- implementations of equation based models which define the behavior of a system or a process under consideration
- various linear and nonlinear equation solvers
- initial and boundary value solvers for differential equations
- numerical continuation methods for nonlinear and differential equations
- computation of normal forms and testing functions
- visualization of results

This patterns are used to design an architecture of the tool under development. The tool will be briefly described in the next section.

1.3 The numerical analysis tool DIANA

The main topic of the present work is the development of a nonlinear analysis tool for chemical engineering models. The principal requirement for the proposed tool DIANA (the name stands for "**D**ynamic s**I**mulation **A**nd **N**onlinear **A**nalysis") is its ability to analyze higher order lumped-parameter or discretized distributed-parameter models. As a modeling front-end for DIANA models the ProMoT modeling tool (TZGG00) is used. Another essential requirement is the usage of open source and freely available numerical codes. The used numerical libraries are distributed with GPL or BSD type licenses. The core libraries are

BLAS (Don02a; Don02b), LAPACK (ABB+99; GV96), UMFPACK (Dav04), and ARPACK (LSY98) for basic linear algebra algorithms, SUNDIALS/Ida (HBG+05) and DASPK (LP99) for the solution of ODE initial value problems, SUNDIALS/Kinsol and NLEQ1S (NW91) for the solution of nonlinear algebraic equations.

The usage diagram of the ProMoT modeling tool with DIANA simulation environment is shown schematically in Figure 1.1. The whole process of numerical analysis can be subdivided into two stages.

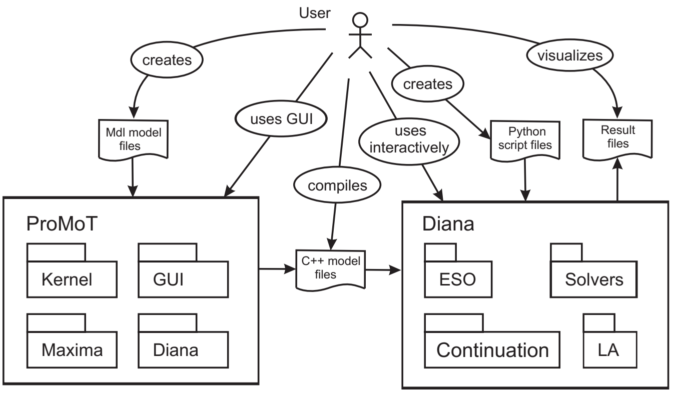


Figure 1.1: ProMoT/DIANA usage diagram

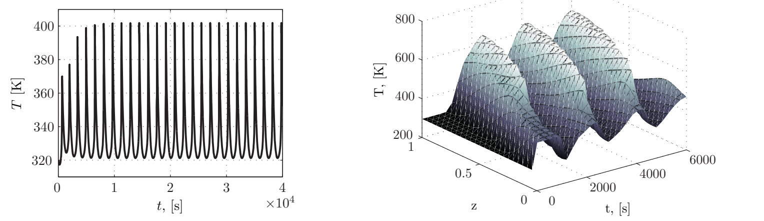
The first stage concerns the creation of a model representation that can be used in numerical computations. The user creates a model in symbolic form, either using the object-oriented model definition language Mdl (TGZG97) or by composing predefined modeling elements from model libraries with the graphical user interface of ProMoT (GKN+03; WAPGK06). The differentiation of the model is performed in the computer algebra system Maxima that is linked with ProMoT. The ProMoT kernel processes the model and translates the model description to Maxima expressions, which are differentiated for a user-defined set of differentiation orders.The model equations as well as the derivatives are translated to C++ source files in GUI or with help of the command line translator `mdl2diana`. The resulting model can be compiled and linked to a shared library, which represents the model for the simulation tool, by the `dianac` script.

The compiled model is used as an input to the simulation environment DIANA. The environment is based on the dynamic object-oriented programming language

Python and inherits the Python command line user interface. Python wrappings for the CAPE-OPEN C++ interfaces by the SWIG (Simplified Wrapper and Interface Generator) software development tool are generated. The wrappings allow to define simulation scenarios interactively or via Python scripts.

Various Python libraries give the possibility to extended simulation capabilities. For example, `NumPy` library is a Python wrapping for the `BLAS/LAPACK` libraries and can be used to apply linear algebra algorithms to results of simulations. The library gives an opportunity to use generic linear algebra algorithms within simulation scripts. On the other hand, GUI libraries, like `PyGTK` or `PyQt`, can be used in online plots during simulations or for a development of a graphical front-end for the simulation results. The developed software has also possibility to save data in a numerical format for the post-processing in other tools, *e.g.*, Matlab.

Some results of the dynamic simulation of a continuous stirred tank reactor (see Section 4.1) or a circulation loop reactor (MKG+99) are presented in Figure 1.2. In Figure 1.3 results of the parameter continuation of steady state and periodic solutions are presented. The solid lines in Figure 1.3 stand for stable solutions, dashed lines stand for unstable solutions, boxes are Hopf bifurcation points, and marked circles are stable periodic solutions.



a) CSTR example b) circulation loop reactor example (MKG$^+$99)

Figure 1.2: Dynamic simulation results in Diana

1.4 Outline of the thesis

The present thesis covers some theoretical aspects of the dynamic simulation and nonlinear analysis, description of software implementation aspects and case studies. A brief outline of the thesis is given in the following.

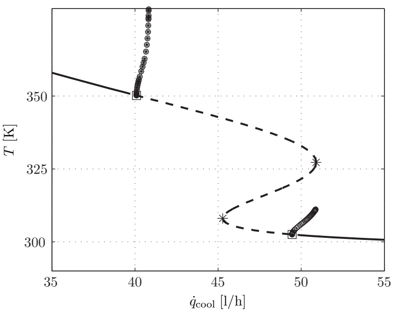


Figure 1.3: Parameter continuation results in DIANA; solid lines are stable steady-state solutions; dashed line is an unstable steady-state solution; circles are periodic solutions: upper branch is stable and lower branch is unstable; upper square is the supercritical Hopf point and lower is the subcritical one; asterisks mark limit points

The parameter continuation and the bifurcation analysis of steady-state points are the focus of Chapter 2. At first, the notion of the dynamical system under treatment is presented. After that follows the basic topic of the chapter — the parameter continuation algorithm and an application to the steady states and their singularities. The chapter highlights the singularity theory and discusses how singularity methods are used in applications. The Lyapunov-Schmidt reduction in the limited context of ordinary differential equations will also be introduced in this chapter. Further the recognition problem and unfolding in a neighborhood of singularities will be presented. After that, the numerical singularity computation and corresponding augmented systems will be discussed. Next topic under discussion are dynamical systems in the presence of a pair purely imaginary eigenvalues. At first the Hopf bifurcation point will be introduced. The numerical computation and corresponding augmented systems for the Hopf bifurcation point will be discussed in the next section. At the end of the chapter the limited case of the Hopf point singularities will be discussed.

Chapter 3 deals with dynamical systems with the presence of periodic solutions. The discussion covers solution with shooting methods of periodic boundary value problems and initialization in the neighborhood of the Hopf bifurcation point. The Recursive Projection Method (SK93) and its applications to various numerical problems are also studied in this chapter.

The three case studies in this thesis form an important part of it — they illustrate how the described methods are used in applications. An important phase in the nonlinear analysis of a problem that is not addressed in this thesis is the model building. The reason for this is that there are no general rules: each problem has its own peculiarities justifying an individual approach. Also it will be assumed that ProMoT and Diana are installed successfully, and that the reader is already familiar with the Python language that will be used to present simulation examples.

Chapter 2

Parameter Continuation Methods and Bifurcations of Equilibria

In the following chapters the main theme of the thesis will be explored: *bifurcation analysis*, the study of possible changes in the structure of solutions of a differential equations system depending on variable parameters. The structural changes of solutions with a parameter change or *bifurcations* can be local, *i.e.*, at such bifurcation point the stability of an equilibrium or fixed point is changed and can be analyzed via the system linearization at the point, or global if the structural changes in the phase space cannot be detected in any small neighborhood, as is the case with local bifurcations. In this chapter numerical algorithms for the bifurcation analysis of the continuous-time dynamical system will be discussed. The analysis of the system will be restricted to the study of the dependence of equilibria or steady-state points on parameters, as well as locating and analyzing their local bifurcations.

Another issue that is covered in the chapter is the singularity analysis of bifurcation varieties. There are numerous publications on applications of singularity analysis to chemical engineering systems. It has been shown in many cases that bifurcation analysis may help to understand the process behavior in greater detail and to improve process design and process operation. The numerical methods for the computation of singularities are also well established. However, there are hardly publications dealing with implementation issues. Especially the automatic generation of the augmented equation systems defining the singularities has hardly been addressed so far in open literature. Currently, this step is done manually by the user in most cases, using some symbolic manipulation tool. In this way, the generation of the augmented system requires a lot of work and insight from the user. Consequently, the application of singularity analysis is currently reserved to specialists with a sound background in nonlinear analysis. On the other hand, the resulting bifurcation diagrams are understandable and instructive for a much broader community. One can expect that bifurcation and singularity analysis will find considerably more applications, if the barriers to use these methods are lower. The purpose of the chapter is to present the nonlinear analysis tools with easy-to-use interfaces and a low entry level to make them comprehensible to non-specialists in the nonlinear analysis.

The chapter is structured as follows:

The first section describes some notions from the dynamical systems and differential-algebraic equation systems. The representation of the dynamical system will be extensively used in the next sections to present nonlinear analysis methods and algorithms. Section 2.2 describes the numerical continuation methods used. The steady-state continuation solver and the linearized stability analysis are topics of Section 2.3.

The special case of the finite-dimensional Lyapunov-Schmidt reduction in a presence of a simple zero eigenvalue is presented in Section 2.4. Section 2.5 summarizes mathematical methods of singularity analysis of real bifurcation varieties. The section gives a brief overview of the main results that are discussed in Chapters I – IV of the book by Golubitsky and Schaeffer (GS85). Another topic is the symbolic differentiation that is widely used in numerical nonlinear analysis and is described in Section 2.6. The higher order derivatives will be used for the computation of bifurcations norm forms or test functions. Section 2.7 discusses the automatic generation of the augmented equation system and presents the singularity analysis solver.

The final three sections are dedicated to the Hopf bifurcation theorem and the related bifurcation point. Section 2.8 discusses the C^L-Hopf Bifurcation Theorem and calculation of the periodic orbit approximation in a neighborhood of the Hopf point. The augmented system and the solver class for the Hopf point calculation in Section 2.9 are presented. Concluding Section 2.10 highlights degenerate Hopf points and test functions that allow to detect such points and perform the parameter continuation.

2.1 Model description

The definition of a simulation model is based on the notion of a dynamical system, the mathematical formalization of the general scientific concept of a deterministic process. The future state of many chemical systems can be predicted to a certain extent by knowing their state and the laws governing their evolution. All possible states of a system can be presented by points of some set or *state space* X of the system. The evolution of a dynamical system is defined as a change of the system state with respect to an independent variable or *time* t. Concerned dynamical systems in the thesis will be time continuous with $t \in \mathbb{R}$. An evaluation law of a dynamical system determines the state $x_t \in X$ of the system at time t, provided the *initial state* $x_0 \in X$ is known. With the time-invariable laws, the behavior of such a system is completely defined by its initial state. An *evaluation operator* of the dynamical system is a map φ^t that is defined in the state space X

$$\varphi^t : X \to X$$

and transforms an initial state $x_0 \in X$ into some state $x_t \in X$ at time $t > 0$:

$$x_t = \varphi^t x_0.$$

It is possible to give a proper definition of a dynamical system.

Definition 2.1. *A* dynamical system *is a pair* $\{X, \varphi^t\}$*, where* X *is a state space and* $\varphi^t : X \to X$ *is a family of evaluation operators satisfying the properties*

$$\varphi^0 = \mathrm{id},$$

where id *is the identity map on* X, id $x \equiv x$ *for all* $x \in X$*, and*

$$\varphi^{t+s} = \varphi^t \circ \varphi^s \quad or \quad \varphi^{t+s}x = \varphi^t(\varphi^s x),$$

for all $x \in X$ *and* $t, s \geqslant 0$*, such that both sides of the last equation are defined.*

In the Diana framework a *differential equation* system is used to define a continuous-time dynamical system. The state space of a system is finite-dimensional $X = \mathbb{R}^n$ with coordinates $(x_1, x_2, \ldots, x_n)$. In the case of differential equations any point x in the state space is determined also by its "velocity" or time derivative $\dot{x} \in \mathbb{R}^n$. The law of evaluation φ^t of the system is given implicitly by a system of *differential-algebraic equations* (DAE)

$$f(t, x, \dot{x}, \nu) = 0, \qquad x(t_0) = x_0, \tag{2.1}$$

where the vector $\nu \in \mathbb{R}^p$ contains time-independent parameters and the vectorvalued function $f : \mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is supposed to be differentiable sufficiently many times or smooth. The case when f has removable discontinuities is not treated in the thesis and topic of the current development in the DIANA framework.

A general system of the differential-algebraic equations in form (2.1) can be characterized by the *differential index* d_i, the minimum integer m, such that the system of equations (2.1) and

$$\begin{aligned} \frac{\mathrm{d}f(t,x,\dot{x},\nu)}{\mathrm{d}t} &= 0 \\ &\cdots \\ \frac{\mathrm{d}^m f(t,x,\dot{x},\nu)}{\mathrm{d}t^m} &= 0 \end{aligned}$$

can be solved for $\dot{x} = \dot{x}(x)$. For alternative definitions of the DAE index, like, geometrical, perturbation, or tractability indices, see (HW96) or (Gea90). For the numerical solution of such systems different reduction techniques are proposed, for example (Pan88; Gea88; MS93).

The DIANA framework allows to solve only differential index-one models. It is not a strong restriction on models, because models with differential index one are most relevant for chemical and biological applications. The pleasing property of index-one systems is the regularity of a DAE, which implies existence and uniqueness results by employing the existence theory of vector fields, as in the following theorem.

Theorem 1. *Let* (2.1) *be a* regular *DAE, let M be the configuration space*[1]*, and let v be the corresponding vector field of this DAE. The vector field v to be of the class C^k for $k > 0$ is assumed. Then, for any* $(t_0; x_0) \in M$ *there exists a solution $x : I \to R^n$ of the DAE on an open interval I containing t_0 with $x(t_0) = x_0$. Moreover, any solution is a mapping of the class C^k and any two solutions x_1 and x_2 with $x_1(t_0) = x_2(t_0)$ are equal on the intersection of their domain.*

Proof. The notion of regularity of DAEs and references to the proof of the above result is presented in (Rei91). $\diamond$

Also throughout the text partial Jacobian matrices $f_x(t,x,\dot{x},\nu)$ and $f_{\dot{x}}(t,x,\dot{x},\nu)$ are assumed to be continuous. Furthermore, the nullspace of $f_{\dot{x}}(t,x,\dot{x},\nu)$ is sup-

[1]the configuration space is the space of possible positions that a physical system may attain and is is typically "half" of a phase space

posed to be invariant of $(x, \dot{x}, \nu)$, *i.e.*,

$$N(t) := \ker f_{\dot{x}}(t, x, \dot{x}, \nu), \tag{2.2}$$

and to vary smoothly with t.

In terms of theorem 1 solutions of the differential equations system (2.1) define a dynamical system. Obviously, the evaluation operator $\varphi^t : \mathbb{R}^n \to \mathbb{R}^n$ with

$$\varphi^t x_0 = x(t, x_0, \nu)$$

in the state space $\mathbb{R}^n$ produces a continuous-time dynamical system $\{\mathbb{R}^n, \varphi^t\}$. The function $x(t) := x(t, x_0, \nu)$ is called a *solution starting at* x_0.

Theorem 1 guarantees existence and uniqueness of the solution $x(t, x_0, \nu)$, but does not give methods to find it. Usually, only in a few simple cases the solution can be found analytically. Computers give possibilities to compute an approximation of a solution curve. For computer numerical methods the dynamical model should be defined and prepared in form of a *simulation model* that have specified interface to communicate with a numerical code. The notion of the simulation model with methods and parameters will be presented in Appendix A.2

2.2 Parameter continuation methods

Continuation methods determine nonsingular solutions of an underdetermined system in the form

$$f(x, \lambda) = 0, \tag{2.3}$$

where $x \in \mathbb{R}^n$ is a state vector, $\lambda \in \nu$ is a free scalar parameter, and $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is a smooth enough function. To simplify this notation further in the section, the vector $\{x, \lambda\}$ will be denoted as $y \in \mathbb{R}^{n+1}$.

If the following conditions are satisfied

(a) $f(x_0, \lambda_0) = 0$, $x_0 \in \mathbb{R}^n$ and $\lambda_0 \in \mathbb{R}$,

(b) the matrix $f_x(x_0, \lambda_0)$ is nonsingular,

(c) f and f_x are smooth near the point $\{x_0, \lambda_0\}$,

then it follows from the Implicit Function Theorem that there exists a unique, smooth function $x(\lambda)$ such that $f(x(\lambda), \lambda) = 0$ and $x(\lambda_0) = x_0$ in the neighborhood of the point $\{x_0, \lambda_0\}$. A solution $y_0 := \{x_0, \lambda_0\}$ of (2.3) is *regular* if the matrix $\mathbb{R}^n \times \mathbb{R}^{n+1}$

$$\{f_x(y_0) \mid f_\lambda(y_0)\}$$

has maximal rank n. If the matrix f_x is singular at the regular point, the point is called a *simple fold* or *limit point*. This situation corresponds to a codim-0 bifurcation point and will be discussed later.

Near the regular point y_0 the Implicit Function Theorem guarantees existence of a unique one-dimensional continuum of solutions. The solutions at the regular point can be parametrized by introducing an additional curve length parameter ζ. With this parameter the solution $y(\zeta)$ must satisfy the equation

$$f(y(\zeta)) = 0, \quad \text{with} \quad y(0) = y_0. \tag{2.4}$$

The numerical solution of the continuation problem (2.4) means computing a sequence of points

$$y^{(0)}, y^{(1)}, \ldots,$$

approximating the solution curve $y(\zeta)$ with a desired accuracy. An initial point $y^{(0)}$ is assumed to be known and equals $\{x_0, \lambda_0\}$. To track the numerical solutions along the arc length parameter ζ, predictor-corrector algorithms are widely used methods, *e.g.,* (Kel77; Sey94). In these methods, the kth step of the continuation starts from an approximation of a solution $y^{(k)}$ and tries to find the next solution point $y^{(k+1)}$:

$$y^{(k)} \rightarrow y^{(k+1)}.$$

With the predictor-corrector methods, that are used in this work, every step is split into two substeps:

$$y^{(k)} \xrightarrow{\text{predictor}} \tilde{y}^{(k+1)} \xrightarrow{\text{corrector}} y^{(k+1)}.$$

The predictor step gives a first estimate of the next point $\tilde{y}^{(k+1)}$ on the solution curve $y(\zeta)$. The corrector locates the point $y^{(k+1)}$ with desired accuracy using the predicted value as an initial guess, and for the corrector the notion of a *parametrization* that allows to calculate the unique point $y^{(k+1)}$ is significant. The distance between two consecutive solution points is called a *step size* $\sigma^{(k)}$. The step size strategy depends on the predictor, the corrector, and the underlying parametrization. In the following, all three components of the predictor-corrector method will be explained in detail.

2.2.1 Predictors

Two approaches have been used here for the predictor step (Figure 2.1), namely:

- the tangent predictor that calculates a tangent vector to the solution curve at the regular point $y^{(k)}$,
- the chord predictor that uses extrapolation formula for the previously computed points.

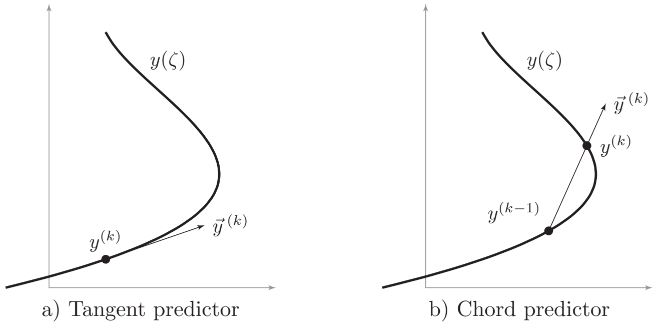


Figure 2.1: Illustration of predictors

Tangent predictor

The tangent predictor calculates a unit tangent vector $\vec{y}^{\,(k)}$ to the solution curve $y(\zeta)$ at the regular point $y^{(k)}$. The tangent vector $\bar{y}^{(k)}$ can be found by solving a linear algebra problem

$$f_y(y^{(k)})\bar{y}^{(k)} = 0 \tag{2.5}$$

that is derived from the differentiation of Equation (2.4) by the arc length parameter ζ. The system of linear algebraic equations (2.5) has a unique solution (up to a scalar multiple) since rank $f_y(y^{(k)}) = n$ by assumption of regularity. To compute the vector $\bar{y}^{(k)}$ from (2.5), one has to extend the linear system by a scalar equation to obtain $n+1 \times n+1$ matrix. The usual way is to fix one variable or a norm of the tangent vector. In this work, the computation of the tangent vector is performed after the corrector iteration and the last Newton iteration matrix in step k is used in (2.5) as the matrix $f_y(y^{(k)})$. The system is extended with a derivative of the parametrization equation, which fixes a norm or a selected variable of the vector $\bar{y}^{(k)}$. This approach reduces numerical costs to compute the tangent vector, because after the corrector iteration the matrix in (2.5) has been already decomposed. Finally, the vector $\bar{y}^{(k)}$is normalized and accepted as the unit tangent vector $\vec{y}^{\,(k)}$.

Chord predictor

The chord predictor is based on the extrapolation of two previously computed points of the solution curve, so at step k the predictor vector is

$$\vec{y}^{\,(k)} = \frac{y^{(k)} - y^{(k-1)}}{||y^{(k)} - y^{(k-1)}||_2}.$$

This approach assumes that at least two successive solution points have been computed. So this predictor type is not suitable for the first step of the continuation algorithm.

2.2.2 Correctors

With the predictor result the initial guess $\tilde{y}^{(k+1)}$ for the corrector step can be defined as

$$\tilde{y}^{(k+1)} := y^{(k)} + \sigma^{(k)} \vec{y}^{\,(k)}.$$

As system (2.4) consists of n equations and $n+1$ unknowns, an auxiliary scalar equation $n(y) = 0$ has to be added that makes the extended system regular. The choice of the auxiliary equation defines the type of parametrization. The following parametrizations are used in this work (Figure 2.2).

Local parametrization

The local parametrization (Sey94; MKMG00) fixes during the corrector step the value of the ith variable. This leads to the parametrization equation

$$n(y) := y_i - \tilde{y}_i^{(k+1)}. \tag{2.6}$$

The index i is determined by the unit tangent vector $\vec{y}^{\,(k)}$ such that

$$|\vec{y}_i^{\,(k)}| = \max(|\vec{y}_1^{\,(k)}|, \ldots, |\vec{y}_{n+1}^{\,(k)}|).$$

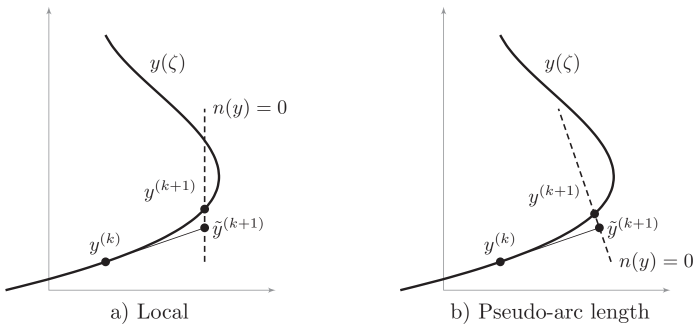


Figure 2.2: Parametrization types

In this case, the variable y_i is locally the most rapidly changing along the curve $y(\zeta)$.

Pseudo-arc length parametrization

The pseudo-arc length parametrization defines a hyperplane passing through the point $\tilde{y}^{(k+1)}$ that is orthogonal to the vector $\vec{y}^{(k)}$:

$$n(y) := \langle y - y^{(k)}, \vec{y}^{(k)} \rangle - \sigma^{(k)}. \tag{2.7}$$

If the curve is regular and the step size $\sigma^{(k)}$ is small enough, one can prove that the Newton iterations for the extended system will converge to a point $y^{(k+1)}$ on the continuation curve, see (Kel77).

Newton iterations

The Gauss-Newton corrector is used to find the exact solution point starting from the approximation of the predictor step. The corrector produces iterations

$$y^{(k+1,i+1)} = y^{(k+1,i)} + \delta^{(i)},$$

where $\delta^{(i)}$ is a Newton update on the ith iteration, and defined as

$$\delta^{(i)} = - \left\{ \begin{array}{c} f_y(y) \\ n_y(y) \end{array} \right\}^{-1} \left\{ \begin{array}{c} f(y) \\ n(y) \end{array} \right\} \Bigg|_{y=y^{(k+1,i)}} . \tag{2.8}$$

The matrix inverse in (2.8) with help of the sparse LU decomposition UMFPACK (Dav04) is computed. This allows to reuse sparsity pattern of the model Jacobian matrix. Also the decomposed matrix of the last Newton iteration is used in the calculation of the tangent predictor. Exit conditions for the Newton iterations are

(a) $||\{f(y^{(k,i)});\ n(y^{(k,i)})\}||_2 < \epsilon_f$.

(b) $||\delta^{(k,i)}||_2/||y^{(k,i)}||_2 < \epsilon_x$ for $||y^{(k,i)}||_2 \neq 0$ or $||\delta^{(k,i)}||_2 < \epsilon_x$ for $||y^{(k,i)}||_2 = 0$.

(c) the maximum number of function evaluations is exceeded.

(d) the maximum number of iterations is exceeded.

For the first two conditions the continuation step is marked as accepted, for the last two the step is rejected and the step size is decreased by the step size control.

2.2.3 The step size control

A step size control uses the rate of convergence of the Newton iteration to adjust the step size σ of the predictor step. In (dHR81) it was shown that a convergence radius cannot be deduced from the previously calculated steps. Because of the negative result regarding of the convergence radii extrapolation, the step size control simply uses the number of the Newton iterations $N^{(k)}$ on the kth step as a parameter for

$$\sigma^{(k+1)} = \xi \sigma^{(k)}, \quad \text{where} \quad \xi = \begin{cases} \xi_1, & N^{(k)} < N_1, \\ 1, & N_1 \leqslant N^{(k)} \leqslant N_2, \\ \xi_2, & N^{(k)} > N_2, \end{cases} \tag{2.9}$$

with scaling factors $\xi_1 > 1$ and $\xi_2 < 1$. The values N_1 and N_2 specify numbers of the Newton iterations for whose the step size will be increased, remains unchanged, or decreased. These parameters have default values $N_1 = 3$, $N_2 = 8$ and depend on the underlying task.

This approach is simple but other more sophisticated variants can be proposed. For example, in Diva the step size control adjust the predictor step size depending on the rate of convergence of the Newton iteration at the corrector step (MKMG00).

2.2.4 The continuation algorithm

The continuation algorithm is based on the above described parts and can be presented as follows:

```
Input: $x_0$, $\lambda_0$, $\sigma^{(0)}$
Output: $y^{(1)}, \dots y^{(k_{max})}$
begin
    $y^{(0)} \leftarrow$ initial_correction($x_0, \lambda_0$)
    $\vec{y}^{(0)} \leftarrow$ predictor($y^{(0)}$)
    $k \leftarrow 0$
    repeat
        $\tilde{y}^{(k+1)} = y^{(k)} + \sigma^{(k)}\vec{y}^{(k)}$
        $y^{(k+1)} \leftarrow$ corrector($\tilde{y}^{(k+1)}$)
        $\vec{y}^{(k+1)} \leftarrow$ predictor($y^{(k+1)}$)
        $\sigma^{(k+1)} \leftarrow$ stepsize($y^{(k+1)}, \sigma^{(k)}$)
        compute_test_functions($y^{(k+1)}$)
        if test condition occurs then
            return ContiOkTestFunction
        end
        $k \leftarrow k + 1$
        if $k \geqslant k_{max}$ then
            return ContiOkMaxStepsMade
        end
    until boundary not achieved ;
    return ContiOkAchievedBoundary
end
```

Algorithm 1: Predictor-corrector parameter continuation

The following notation for algorithms will be used throughout the text. Keywords are designated by the **bold font**, function calls are presented with the `typewriter font`, the **bold sans-serif font** denotes constants, and *italic type* represents variables or shortened texts in algorithms. The **Input** and **Output** sections contain, respectively, arguments and results of an algorithm. The main algorithm body is enclosed in the keywords pair **begin**/**end**. For the algorithm control flow usual statements for loops or conditions, like, **repeat/until** or **if/then/end**, are used. The symbol $\leftarrow$ stands for the assignment statement, which can be parallel, *e.g.,* $a, b \leftarrow 1, 2$.

The main parts of the above algorithm has been described above in this section. The function `compute_test_functions` has abstract definition and defines an evaluation of task-defined *test functions* that are computed along the solution curve. This function closely related to the continuation sub-task and will be described in following sections.

The realization of the parameter continuation solver base class `DianaContinuation` is described in appendix section A.4. The description of the solver contains methods and parameters definitions. As an example of the application of the elaborated continuation methods can serve the steady-state continuation solver that is described in the next section.

2.3 Steady-state continuation and the linearized stability analysis

A steady state of the dynamical system is characterized by the vanishing time derivatives vector $\dot{x}$ in Equation (2.1). In addition, the differential-algebraic equations system is implied to be autonomous, so it does not depend on the variable t. Using the above assumptions, the task under study will have the form of a nonlinear algebraic system

$$f(x, \nu) = 0, \qquad f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n, \tag{2.10}$$

where f is a sufficiently smooth function. This system can be used directly in the algorithm described in Section 2.2 for the steady-state continuation.

Information about stability of a steady-state point x_0 of the dynamical system (2.1) can be found from the linearized system of equations in the neighborhood of the point. This important result in the study of dynamical systems is stated in the following theorem.

Theorem 2 (Local Hartman-Grobman Theorem for Flows)**.** *Let*

$$\varphi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$$

be the (local) flow generated by an autonomous equation

$$\dot{x} = f(x)$$

with a hyperbolic equilibrium point x_0. Let A denote the linearization of f at point x_0. Then there are neighborhoods $\mathcal{U}$ and $\mathcal{V}$ of x_0 and a homeomorphism

$$h : \mathcal{U} \rightarrow \mathcal{V}$$

such that

$$\varphi(h(x), t) = h(x_0 + e^{tA}(x - x_0))$$

whenever $x \in \mathcal{U}$ and $x_0 + e^{tA}(x - x_0) \in \mathcal{U}$ or, in a neighborhood $\mathcal{U}$ of x_0, φ is topologically conjugate to its linearization.

Proof. The proof of the theorem is based on a global Hartman-Grobman theorem for maps (Gro59; Har60). ◇

The linearization of an autonomous differential-algebraic equation at a hyperbolic equilibrium point x_0 results in the following generalized eigenvalue problem

$$f_{\dot{x}}(x_0, 0, \nu)V\Lambda + f_x(x_0, 0, \nu)V = 0. \tag{2.11}$$

where the diagonal matrix Λ contains generalized eigenvalues Λ_i and V is the corresponding eigenspace such that

$$\varphi(h(x), t) = h(x_0 + Ve^{t\Lambda}(x - x_0))$$

with h defined in Theorem 2.

The equilibrium point x_0 of a dynamical system is called *hyperbolic* if none of the real parts of the eigenvalues of (2.11) are equal to zero. The steady state of the dynamical system in the implicit DAE form (2.1) is locally stable at the hyperbolic equilibrium if the real parts of all eigenvalues of the linearized system are positive; it is unstable if at least one real part is negative. It should also be noted that the stability results in Diana, due to the implicit form of the differential-algebraic equation (2.1), differ from standard results in the linearized stability that assumes local stability for negative real parts of all eigenvalues of A and loss of stability if at least a one eigenvalue has positive real part.

The case of a vanishing real part of at least one eigenvalue is called a *critical point*. The presence and behavior of such critical points is of great interest for understanding the dynamics of the system, as critical points indicate stability changes and bifurcations.

A solution of the generalized eigenvalue problem (2.11) in the solver implementation with the QZ algorithm (`ggev` function in the LAPACK library (ABB+99)) or with the implicitly restarted Arnoldi iteration (`naupd` function in the ARPACK library (LSY98)) is performed. The steady state continuation solver computes generalized eigenvalues along a continuation curve and allows to detect a sign change in the real parts of the eigenvalues (Figure 2.3). The limit point bifurcation in

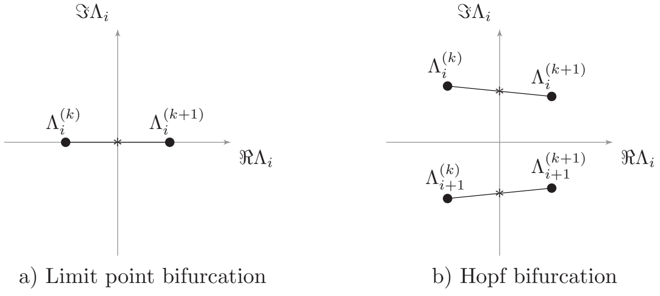


Figure 2.3: Local bifurcations of an equilibrium

Figure 2.3a is characterized by a simple zero eigenvalue Λ_i or singularity of f_x. With the second bifurcation a pair of complex conjugate eigenvalues of the linearization (2.11) around the fixed point cross the imaginary axis of the complex plane. The point has the name *Hopf* or *Andronov-Hopf bifurcation* and is characterized by a stability change and, under reasonably generic assumptions about the dynamical system, a periodic orbit branching from the fixed point.

The implementation of the steady-state parameter continuation solver class **SteadyStateContinuation** with methods and parameters description is described in Appendix A.4.1. Initial values for the continuation solver are taken from variables and parameters of an ESO model instance. These values can contain, for example, results of a dynamic simulation of the model or previous parameter continuation results.

Further analysis in this chapter will be devoted to the analysis of non-hyperbolic equilibrium points. In Sections 2.4 and 2.5 the theoretical background for the singularity analysis of limit point varieties will be summarized. Section 2.7 presents the numerical computation and the continuation of such varieties and a solver class.

2.4 The Lyapunov-Schmidt reduction

In engineering applications it is quite important to determine stability domains of the system. One of the possible scenarios for the stability change is a crossing of the imaginary axis by a real eigenvalue. The scenario corresponds to the fold or limit bifurcation point where an equilibrium point may lose stability. The parameter dependency of the fold points can be analyzed via reduction of the system to a single scalar equation and computing solutions of this equation. In the following section the finite-dimensional Lyapunov-Schmidt reduction will be presented (GS85, I.3).

Consider the nonlinear system

$$f(x,\nu)=0, \qquad f:\mathbb{R}^n\times\mathbb{R}^p\to\mathbb{R}^n \tag{2.12}$$

that defines the equilibrium point curve of the underlying differential equation (2.1). Let at the solution point x_0, ν_0 of the system (2.12) the Jacobian matrix $A := f_x(x_0,\nu_0)$ have rank $n-1$. In the case with the Lyapunov-Schmidt reduction a scalar equation may be defined as

$$g(z,\nu)=0, \qquad g:\mathbb{R}\times\mathbb{R}^p\to\mathbb{R} \tag{2.13}$$

that has locally one-to-one correspondent solutions to the original full system.

The reduction procedure assumes a definition of $\mathbb{R}^n$ orthogonal space splittings

$$\begin{aligned}\mathbb{R}^n &= \ker A\oplus M,\\ \mathbb{R}^n &= N\oplus \operatorname{range} A\end{aligned} \tag{2.14}$$

with dimensions $\dim\operatorname{range} A = n-1$, $\dim\ker A = 1$, so that $\dim M = n-1$, and $\dim N = 1$. The projection E of $\mathbb{R}^n$ onto $\operatorname{range} A$ splits the original system (2.12) into an equivalent pair of equations

$$\begin{aligned}Ef(x,\nu) &= 0,\\ (I-E)f(x,\nu) &= 0.\end{aligned} \tag{2.15}$$

The vector x also can be decomposed in form $x=v+w$ where $v\in\ker A$ and $w\in M$.

So (2.15) has the form

$$\begin{aligned} Ef(v+w,\nu) &= 0, \\ (I-E)f(v+w,\nu) &= 0. \end{aligned} \tag{2.16}$$

For the first equation of the Implicit Function Theorem a mapping $W(v,\nu)$ can be found that satisfies near the point $x_0 = v_0 + w_0$, ν_0

$$Ef(v+W(v,\nu),\nu) \equiv 0, \quad W(v_0,\nu_0) = w_0. \tag{2.17}$$

In fact, differentiation of the first equation in (2.16) with respect to the w variable at the solution point x_0, ν_0 produces a linear map

$$EA : M \to \text{range}\, A \tag{2.18}$$

that is invertible. Now, the reduced map $\phi : \ker A \times \mathbb{R}^p \to N$ can be obtained from the second equation in (2.16)

$$\phi(v,\nu) = (I-E)f(v+W(v,\nu),\nu). \tag{2.19}$$

The reduced function ϕ may be calculated in numerical applications in an appropriate basis for subspaces of $\mathbb{R}^n$, namely, $\ker A$ and N. With unit-length basis vectors $v_0 \in \ker A$ and $v_0^* \in N$ system (2.19) can be defined as a scalar function

$$g(z,\nu) = \langle v_0^*, \phi(zv_0,\nu)\rangle, \tag{2.20}$$

where $\langle .,. \rangle$ is the Euclidean inner product. The function (2.20) does not depend on the projection E, indeed

$$\begin{aligned} g(z,\nu) &= \langle v_0^*, (I-E)f(zv_0 + W(zv_0,\nu),\nu)\rangle \\ &= \langle v_0^*, f(zv_0 + W(zv_0,\nu),\nu)\rangle \end{aligned} \tag{2.21}$$

due to orthogonality of the vector v_0^* to the range A. The zeros of g are in one-to-one correspondence with ϕ and with solutions of $f(x,\nu) = 0$ independently from the choice of the vectors $v_0 \in \ker A$ and $v_0^* \in N$. This result is proved in (GS85, A2).

The augmented system that specifies a numerical computation of the vectors v_0 and v_0^* will be presented in Section 2.7.

For the analysis of zeros of $g(z,\nu)$, a computation of derivatives with respect to z and ν is required. The derivatives computation of g is based on the higher-order directional derivatives of f that are introduced as follows

$$(d^k f)_{x,\nu}(v_1,\ldots,v_k) = \frac{\partial}{\partial t_1}\cdots\frac{\partial}{\partial t_k} f\left(x+\sum_{i=1}^{k} t_i v_i, \nu\right)\Bigg|_{t_1=\cdots=t_k=0}. \tag{2.22}$$

Numerically such derivatives are calculated with symbolically computed higher-order derivatives of f (2.32), for example, if $k=3$

$$(d^3 f)_{x,\nu}(u,v,w) = \sum_{i,j,l=1}^{n} \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_l}(x,\nu) u_i v_i w_j.$$

It also should be noted that the differentiation of (2.17) with respect to z results after the substitution $v = zv_0$ at the point x_0, ν_0

$$0 = Ef_x(zv_0 + W(z,\nu),\nu)\cdot(v_0 + W_z(z,\nu)) = EAW_z(z,\nu) = W_z(z,\nu),$$

with $EAv_0 = 0$ and the invertible linear map EA.

Applying the chain rule to (2.21) yields following expressions for the derivatives of g with respect to the variable z and a parameter $\lambda \in \nu$

$$\begin{aligned}
g_z &= \langle v_0^*, d\,f(v_0+W_z)\rangle,\\
g_{zz} &= \langle v_0^*, d^2 f(v_0+W_z, v_0+W_z) + d\,f(v_0+W_{zz})\rangle,\\
g_{zzz} &= \langle v_0^*, d^3 f(v_0+W_z, v_0+W_z, v_0+W_z)\\
&\quad +3d^2 f(v_0+W_z, W_{zz}) + d\,f(v_0+W_{zzz})\rangle,\\
g_\lambda &= \langle v_0^*, f_\lambda + d\,f(W_\lambda)\rangle,\\
g_{z\lambda} &= \langle v_0^*, d\,f_\lambda(v_0+W_z) + d^2 f(v_0+W_z, W_\lambda) + d\,f(W_{z\lambda})\rangle.
\end{aligned} \tag{2.23}$$

Recalling that v_0^* is orthogonal to range A and $W_z(z_0,\nu_0)=0$, the formulas (2.23) at the point x_0, ν_0 become

$$\begin{aligned}
g_z &= 0,\\
g_{zz} &= \langle v_0^*, d^2 f(v_0,v_0)\rangle,\\
g_{zzz} &= \langle v_0^*, d^3 f(v_0,v_0,v_0) + 3d^2 f(v_0, W_{zz})\rangle,\\
g_\lambda &= \langle v_0^*, f_\lambda\rangle,\\
g_{z\lambda} &= \langle v_0^*, d\,f_\lambda(v_0) + d^2 f(v_0, W_\lambda)\rangle
\end{aligned} \tag{2.24}$$

for the higher-order derivatives $d^2 f$, $d^3 f$, f_λ, and $d\, f_\lambda$ that are evaluated at the point x_0, ν_0. Unknown derivative W_{zz} can be obtained via double differentiation of (2.17) with respect to z

$$E \cdot d\, f_{x,\nu}(W_{zz}) + E \cdot d^2 f_{x,\nu}(v_0 + W_z, v_0 + W_z) = 0$$

or at the point x_0, ν_0

$$W_{zz} = -A^{-1} \cdot E \cdot d^2 f_{x_0,\nu_0}(v_0, v_0).$$

Similar, differentiation of (2.17) with respect to λ yields to

$$W_\nu = -A^{-1} \cdot E \cdot f_\lambda.$$

The next section shows some theoretical results of the singularity theory and applications to the bifurcations analysis of the above defined reduced function g.

2.5 The singularity recognition and unfolding

In this section singularities of a scalar smooth function

$$g(z, \nu) : \mathbb{R} \times \mathbb{R}^p \to \mathbb{R} \tag{2.25}$$

and their application to the bifurcation theory of DAEs will be studied. By a *bifurcation* a change in the number of solutions of the equation $g(z, \nu) = 0$ as parameters ν vary will be defined. If equation (2.25) represents, for example, a reduced DAE system in the neighborhood of a fold point, then the bifurcation analysis results of the scalar equation will show qualitative changes of equilibrium points in the phase portrait of the underlying DAE system.

Without loss of generality, let the function g have a zero at the origin $g(0, 0) = 0$. The trivial case $g_z(0, 0) \neq 0$ is treated by the Implicit Function Theorem and (2.25) can be uniquely solved for z as a function of ν at some neighborhood of the origin. A point for which $g_z(0, 0) = 0$ is called a *singularity* and will be the main topic of the section. At such points $n_g(\nu)$, the number of solutions of $g(z, \nu) = 0$, discontinuously changes the value. To the rest of the section $\lambda \in \nu$ will be the selected scalar parameter under consideration of the singularity analysis.

To classify the bifurcation diagrams of equation (2.25) an equivalence relation should be introduced. Two smooth functions g, $h : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ in a sufficiently small neighborhood of the origin are equivalent if there exist functions $Z(z, \lambda)$, $\Lambda(\lambda)$ and $S(z, \lambda)$ such that the relation

$$h(z, \lambda) = S(z, \lambda)g(Z(z, \lambda), \Lambda(\lambda)) \tag{2.26}$$

holds around the origin. Moreover, the following requirements

$$S(z, \lambda) > 0, \quad Z_z(z, \lambda) > 0, \quad \Lambda'(\lambda) > 0 \text{ and } Z(0, 0) = \Lambda(0) = 0$$

should be satisfied. If g and h are equivalent, then their multiplicity functions are related as follows

$$n_g(\Lambda(\lambda)) = n_h(\lambda). \tag{2.27}$$

With the defined equivalence relation the following *recognition problem* can be formulated. For a given function g, which has the simplest form, find all the functions h equivalent to g and define criteria that characterize such functions. The simplest function g is called a *normal form* and represents a class of equivalent bifurcation problems. The solution of the recognition problem by singularity theory methods is based on a finite list of terms in the Taylor series of h such that the question of equivalence is determined only by the values of the derivatives of h on this list, and all other terms can be ignored. The monomials $z^k\lambda^m$ in the Taylor series are divided into three classes: low order terms that do not appear in the series of h, higher-order terms that have no influence on the equivalence, and non-zero intermediate-order terms. In numerical applications degenerate singular points will be detected by vanishing low order terms and non-vanishing intermediate-order terms.

After the recognition of the equivalence class for the function g in terms of the singularity theory, another type of problem may be formulated. This problem concerns a small perturbation of the function g. Such perturbations may cause significant changes in the solutions bifurcation diagram of the function. The study of such changes is termed “imperfect bifurcation”.

Under a *versal unfolding* of a smooth function $g(z, \lambda)$ a parametrized family of functions will be understood

$$G(z, \lambda, \alpha), \tag{2.28}$$

where α are unfolding k parameters ($\alpha \subseteq \nu$), satisfy the following two conditions:

(a) $G(z, \lambda, 0) = g(z, \lambda)$,

(b) any sufficiently small perturbation of g is equivalent to $G(z, \lambda, \alpha)$ in the neighborhood of the origin.

If the number of additional parameters is irreducible, then G is called a *universal unfolding*. The minimal number k of parameters α, needed for the description of all perturbations, is called the *codimension* of g. In this work, the definition of the codimension based on the number of unfolding parameters is used (GS85). Some authors use an alternative definition of the codimension based on the number of vanishing derivatives (GM97) or the number of independent conditions determining the bifurcation (Kuz04). The second definition leads to a value of the codimension larger by one compared to the definition used in this work, *i.e.*, a limit point is a codimension-0 singularity in this work, but a codimension-1 singularity according to the second definition.

The notion of unfolding has the following aspects which are important for applications. Equation

$$G(z, \lambda, \alpha) = 0 \tag{2.29}$$

appears in many mathematical models of chemical engineering problems. It always neglects some physical effects, many of those are beyond the control of the person performing the experiment. So, a more accurate description of the physical system would lead to

$$G(z, \lambda, \alpha) + p(z, \lambda, \alpha) = 0, \tag{2.30}$$

where p is a small perturbation representing what the model neglects. Thus, it only remains to require that equation (2.29) has bifurcation diagrams equivalent to equation (2.30) in a neighborhood of a subset of interesting parameters $\alpha_0 \subseteq \alpha$. This is satisfied when $G(z, \lambda, \alpha)$ is a versal (or universal) unfolding of $G(z, \lambda, \alpha_0)$ and this fact has two meaningful consequences for the model. First, it ensures that the model describes all possible phenomena. Secondly, it is possible to find which of the parameters (or algebraic expressions formed from them) are important for the universal unfolding. The necessary condition for $G(z, \lambda, \alpha)$ to be a versal unfolding is that the codimension of $G(z, \lambda, \alpha_0)$ is not greater than the number of parameters α_0. On the other hand, when the codimension of $G(z, \lambda, \alpha_0)$ exceeds the number of auxiliary parameters, the model should be approached with caution. There are three possibilities for an explanation. Either the model is not able to describe all phenomena which can occur, or the problem exhibits a symmetry which makes the codimension smaller, or in the universal unfolding of $G(z, \lambda, \alpha_0)$ there are so-called modal parameters (moduli) which do not influence the bifurcation diagrams from

Normal form	codim	Name
$\varepsilon z^2 + \delta\lambda$	0	Limit point
$\varepsilon(z^2 - \lambda^2)$	1	Simple bifurcation
$\varepsilon(z^2 + \lambda^2)$	1	Isola center
$\varepsilon z^3 + \delta\lambda$	1	Hysteresis
$\varepsilon z^2 + \delta\lambda^3$	2	Asymmetric cusp
$\varepsilon z^3 + \delta\lambda z$	2	Pitchfork
$\varepsilon z^4 + \delta\lambda$	2	Quartic fold
$\varepsilon z^2 + \delta\lambda^4$	3	
$\varepsilon z^3 + \delta\lambda^2$	3	Winged cusp
$\varepsilon z^4 + \delta\lambda z$	3	
$\varepsilon z^5 + \delta\lambda$	3	

Table 2.1: Normal forms for singularities of codim $g \leqslant 3$

the topological point of view.

The main result of the singularity theory for singularities with codimension three or less is stated in the following theorem.

Theorem 3 (The Classification Theorem (GS85, IV.2)). *Let $g(z, \lambda) : \mathbb{R}^2 \to \mathbb{R}$ be a function that is defined in a sufficiently small neighborhood of the origin, satisfying $g = g_z = 0$ at $(0, 0)$. If* codim *$g \leqslant 3$, then g is equivalent to one of the following bifurcation problems listed in table 2.1.*

Parameters ε and δ of the normal forms in table 2.1 equal ± 1, so all possible signs of monomials in the normal forms are considered. The origin point of the normal forms in table 2.1 presents different type of bifurcations, like, simple bifurcation (either transcritical or isola), hysteresis, pitchfork, and winged cusp points. These points are so-called *organizing centers* that are associated with a distinguished set of values for the parameters such that all possible different qualitative behaviors occur for parameter values in a small neighborhood of the distinguished values. Such kind of points exhibit the most singular behavior and pseudo-global results often may be obtained by the application of local analysis near the organizing center.

Applications of the theoretical results above, in numerical computations, will be shown in the next section.

2.6 Symbolic differentiation

The most commonly used approach for computing derivatives are finite difference schemes with different orders. The numerical error for derivative of order i calculated by finite differences can be described as a sum of the round-off and the truncation errors in the simplest case (for example (Hig96)):

$$err_i = a_i \frac{\delta_m}{\epsilon^i} + b_i \epsilon, \tag{2.31}$$

where a_i and b_i are factors that depend on the mapping $f(x, \dot{x}, \nu)$ and δ is the machine unit round-off. In Figure 2.4 (factors $a_i = b_i = 1$ and $\delta_m = 10^{-16}$) it is possible to see that with an increase of the order of differentiation the absolute numerical error increases, whereas the interval of the permitted ϵ values for a required tolerance relative to the optimum value decreases. This means that for higher order derivatives it becomes increasingly difficult to choose a suitable value for ϵ. The numerical approximation of the higher order derivatives may become inaccurate, if ϵ differs slightly from the optimal value (which usually is not known). For this reason, it was decided against the usage of finite difference approximation in this work.

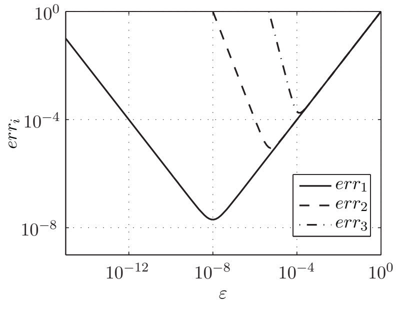


Figure 2.4: Numerical error of derivatives calculated by the finite differences with the first order approximation

To avoid the described numerical problems, automatic or symbolic differentiation of f can be used. For the problems considered here high order derivatives in sparse form are required. Such derivatives are used, for example, to compute derivatives of the function g after the Lyapunov-Schmidt reduction in (2.24). Thorough analysis of existing tools shows that no automatic differentiation code exists that can handle this sort of problem. Therefore, an approach based on symbolic differentiation is proposed in the following. The algorithm is able to differentiate via computer algebra system Maxima (FdSMY04) any mathematical expression that can be present in ProMoT models.

The differentiation approach for the summation expressions is based on the assumptions that follow from the chain rule and the differentiation δ-rule proposed in (Wan94). The chain rule requires that all derivative entries with the same indices should be summed. Differentiation of the summations can be described with the Kronecker symbol, defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases}$$

For single sums of the form

$$\sum_{i=\Delta} S(x_i)$$

the derivative by the variable x_j can be presented as $S'(x_i)\delta_{ij}$ with loop indices $i, j \in \Delta$. Multiple summations

$$\sum_{i_1=\Delta_1} \cdots \sum_{i_n=\Delta_n} S(x_{i_1,\ldots,i_n})$$

can be differentiated by the variable $x_{j_1,\ldots,j_n}$ in the same manner and the result will be

$$S'(x_{i_1,\ldots,i_n})\delta_{i_1 j_1} \ldots \delta_{i_n j_n}$$

for loops indices $i_1, j_1 \in \Delta_1, \ldots, i_n, j_n \in \Delta_n$. The order of loops can be arbitrary due to the commutative property of the summation.

Differentiation of conditional and summation expressions are performed by differentiation of conditional branches. The treatment of conditionals and summation expressions is illustrated by the following example:

$$\begin{aligned} f1(x)_i &:= x_1 + x_i + \left\{ \begin{array}{ll} \exp(x_{n-i+1}), & x_1 \geqslant 0 \\ \sin(x_i), & x_1 < 0 \end{array} \right\}, \quad \text{for } i = 1 \ldots n, \\ f2(x, y) &:= x_1 + \textstyle\sum_{i=1}^{n} x_i + y. \end{aligned}$$

The symbolic differentiation in Maxima produces:

$$\begin{aligned}
\partial f1(x)_i/\partial x_1 &= 1 \\
\partial f1(x)_i/\partial x_i &= 1 + \left\{ \begin{array}{ll} 0, & x_1 \geqslant 0 \\ \cos(x_i), & x_1 < 0 \end{array} \right\}, \\
\partial f1(x)_i/\partial x_{n-i+1} &= \left\{ \begin{array}{ll} \exp(x_{n-i+1}), & x_1 \geqslant 0 \\ 0, & x_1 < 0 \end{array} \right\},
\end{aligned}$$

and

$$\partial f2(x,y)/\partial x_1 = 1, \quad \partial f2(x,y)/\partial x_i = 1, \quad \partial f2(x,y)/\partial y = 1.$$

Finally, the resulting matrix $\partial\{f1, f2\}/\partial\{x, y\}$ is formed by sorting all produced symbolic expressions into an initially empty matrix and taking into account the limits of the index i:

$$\begin{array}{cccccc}
1 + \frac{\partial f1(x)_i}{\partial x_i}|_{i=1} & 0 & 0 \ \dots & 0 & \frac{\partial f1(x)_i}{\partial x_{n-i+1}}|_{i=1} & 0 \\
1 & 1 + \frac{\partial f1(x)_i}{\partial x_i}|_{i=2} & 0 \ \dots & \frac{\partial f1(x)_i}{\partial x_{n-i+1}}|_{i=2} & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 + \frac{\partial f1(x)_i}{\partial x_{n-i+1}}|_{i=n} & 0 & 0 \ \dots & 0 & 1 + \frac{\partial f1(x)_i}{\partial x_i}|_{i=n} & 0 \\
1+1 & 1 & 1 \ \dots & 1 & 1 & 1
\end{array}$$

By applying the described procedure recursively, higher order derivatives of the model equations with respect to state variables, state derivative or parameters can be generated. The result are expressions

$$\frac{\partial^{(k+l+m)} f(x, \dot{x}, \nu)}{\partial x^{(k)} \partial \dot{x}^{(l)} \partial \nu^{(m)}}, \quad k, l, m \geqslant 0 \tag{2.32}$$

with a $k+l+m+1$ element indices tuple $(i_1, \dots, i_{k+l+m+1})$. The obtained higher order derivatives will be used for the singularity analysis of the reduced test function in the numerical realization of the singularity analysis solver in Section 2.5.

2.7 The numerical singularity computation

The problem of computing higher order singularities can be divided into two subproblems, namely, the generation of an augmented equation system defining the considered singularity, and the numerical solution of the nonlinear system. The last problem is discussed in Section 2.2 and will not be treated here.

The solution of the first subproblem requires extensive symbolic manipulations. Especially for complex high-order chemical process models, the manipulations can no longer be done manually, but a computer tool is highly desirable that generates the augmented system automatically.

The Lyapunov-Schmidt reduction, which is presented in Section 2.4, requires an augmentation of the initial system (2.10). In this work, the approach proposed by Kunkel is used (Kun91). The augmented system of choice is

$$\begin{cases} f(x,\nu) &= 0, \\ f_x(x,\nu)v_0 - \beta v_0^* &= 0, \\ f_x^T(x,\nu)v_0^* - \gamma v_0 &= 0, \\ ||v_0||_2 &= 1, \\ ||v_0^*||_2 &= 1, \end{cases} \tag{2.33}$$

with the singular values β and γ, the eigenvector v_0 and the adjoint eigenvector v_0^* that correspond to the zero eigenvalue. The augmented system has $3n+2$ equations.

The definition of a singularity of codimension k leads to $k+1$ additional equations, which fix the parameter vector $\alpha \in \mathbb{R}^{k+1}$, which is a subset of the model parameters ν. The $k+1$ additional equations can be described by using the following test functions, which result from the Lyapunov-Schmidt reduction (2.24):

$$\begin{aligned} g_z &= \langle v_0^*, d\, f(v_0)\rangle, \\ g_{zz} &= \langle v_0^*, d^2 f(v_0, v_0)\rangle, \\ g_\lambda &= \langle v_0^*, f_\lambda\rangle, \\ g_{z\lambda} &= \langle v_0^*, d\, f_\lambda(v_0) - d^2 f(v_0, f_x^{-1} f_\lambda)\rangle, \end{aligned} \tag{2.34}$$

where $\lambda \in \nu$ is the parameter that is used in the singularity normal form definition and does not depend on the choice of the parameters subset α, f_λ is a derivative of f by the parameter λ, $d^k f$ is defined by (2.22), and the inner product $\langle x, y\rangle$ stands for the vector product $x^T y$.

Because the Jacobian matrix f_x is singular for $g_z = 0$, the vector $f_x^{-1} f_\lambda$ in the test function $g_{z\lambda}$ is calculated by an iterative least squares solver LSQR (PS82).

Based on the test functions $g_z, \ldots, g_{z\lambda}$ different conditions for singularities can be defined. One obtains the following singularity conditions:

- limit point bifurcation (codimension 0, $\alpha \in \mathbb{R}^1$):
  $g_z = 0$, equivalent to the normal form $\pm z^2 \pm \lambda$

- isola or simple transcritical bifurcation (codimension 1, $\alpha \in \mathbb{R}^2$):
  $g_z = g_\lambda = 0$, equivalent to the normal form $\pm z^2 \pm \lambda^2$

- hysteresis (codimension 1, $\alpha \in \mathbb{R}^2$):
  $g_z = g_{zz} = 0$, equivalent to the normal form $\pm z^3 \pm \lambda$

- pitchfork (codimension 2, $\alpha \in \mathbb{R}^3$):
  $g_z = g_{zz} = g_\lambda = 0$, equivalent to the normal form $\pm z^3 \pm \lambda z$

- winged cusp (codimension 3, $\alpha \in \mathbb{R}^4$):
  $g_z = g_{zz} = g_\lambda = g_{z\lambda} = 0$, equivalent to the normal form $\pm z^3 \pm \lambda^2$

The corresponding flow chart in Figure 2.5 shows an analysis sequence for singularities with codim $\leqslant 3$. Nodes in the flow chart are normal forms that may be equivalent to the reduced model equation under study. Edges show directions of the recognition process and corresponding zeros of the test functions (2.34). In

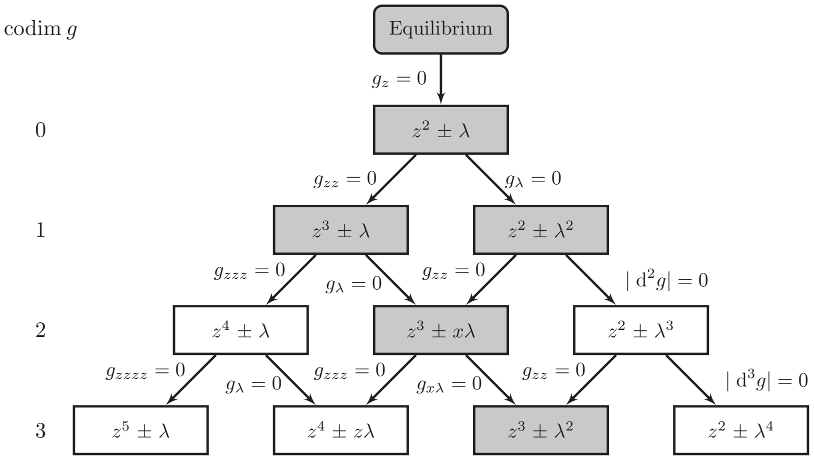


Figure 2.5: A flow chart for the recognition problem of singularities of codimension $\leqslant 3$ (shaded blocks have been implemented)

Figure 2.5 the conditions $|\,\mathrm{d}^2 g| = 0$ and $|\,\mathrm{d}^3 g| = 0$ specify points where matrices

$$\mathrm{d}^2 g = \begin{pmatrix} g_\lambda & g_{\lambda z} \\ g_\alpha & g_{\alpha z} \end{pmatrix}$$

and

$$\mathrm{d}^3 g = \begin{pmatrix} 0 & g_{zz} & g_{z\lambda} \\ g_\alpha & g_{\alpha z} & g_{\alpha\lambda} \\ g_\beta & g_{\beta z} & g_{\beta\lambda} \end{pmatrix}$$

are singular (GS85, Table 3.2).

The numerical analysis starts from an equilibrium point where the test function g_z vanishes. This situation corresponds to codimension-0 limit point and can be detected during a steady-state continuation by a real eigenvalue that crosses the imaginary axis. With the continuation of a limit point's branch, points with vanishing g_{zz} and g_λ can be found. These points can be used as initial points for the continuation hysteresis or simple bifurcation varieties, respectively. At the pitchfork point the derivatives g_{zz} and g_λ are simultaneously equal to zero. On the pitchfork variety with the derivative $g_{z\lambda}$ the most degenerate winged cusp point can be detected. Such organizing point shows diversity in nonlinear phenomena in a small enough neighborhood. Detection of the other five degenerate points, that are equivalent to the normal forms $z^5 \pm \lambda$, $z^4 \pm z\lambda$, $z^2 \pm \lambda^4$, $z^4 \pm \lambda$, and $z^2 \pm \lambda^3$, requires implementation of the test function g_{zzz} and g_{zzzz}.

The implementation of the singularity analysis solver `SingAnalyser` with methods and parameters is described in Appendix A.4.2.

2.8 The Hopf bifurcation

The second type of equilibrium bifurcation, which has been presented in Figure 2.3b, is the Hopf point. It is a local bifurcation in which a fixed point of an autonomous dynamical system

$$f(x, \dot{x}, \nu) = 0 \tag{2.35}$$

loses stability as a pair of complex conjugate eigenvalues of the linearization around the fixed point cross the imaginary axis of the complex plane. The existence and the properties of a small amplitude limit cycle branching from the fixed point can be described by the Hopf bifurcation theorem. In the following theorem, without loss of generality, the equilibrium point is assumed at the origin and the critical value of the bifurcation parameter κ is 0.

Theorem 4 (C^L-Hopf Bifurcation Theorem (HKW81))**.** *If*

(a) *$f(0,0,\kappa) = 0$ for a scalar parameter $\kappa \in \nu$ in an open interval containing 0, and $0 \in \mathbb{R}^n$ is an isolated stationary point of f,*

(b) *all partial derivatives of the components $f^{(l)}$ of the vector f of orders $l \leqslant L+2$ ($L \geqslant 2$) exist and are continuous in x, $\dot{x}$ and κ in a neighborhood of $(0,0,0)$ in $\mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$,*

(c) *$A(\kappa) = f_x(0,0,\kappa)$ and $\dot{A}(\kappa) = f_{\dot{x}}(0,0,\kappa)$ have a pair of complex conjugate generalized eigenvalues Λ_1 and $\bar{\Lambda}_1$ such that*

$$\Lambda(\kappa) = \alpha(\kappa) + i\omega(\kappa),$$

where

$$\omega(0) = \omega_0 > 0, \quad \alpha(0) = 0, \quad \alpha'(0) \neq 0,$$

(d) *the remaining generalized eigenvalues have strictly negative real parts,*

then the system (2.35) has a family of periodic solutions: there exist an $\epsilon_P > 0$ and a C^{L+1}-function

$$\kappa(\epsilon) = \sum_{i=1}^{L/2} \kappa_{2i}\epsilon^{2i} + O(\epsilon^{L+1}) \qquad (0 < \epsilon < \epsilon_P)$$

such that for each $\epsilon \in (0, \epsilon_P)$ there exists a periodic solution $p_\epsilon(t)$, occurring for $\kappa = \kappa(\epsilon)$. There is a neighborhood $\mathcal{U}$ of $x = 0$ and an open interval $\mathscr{I}$ containing 0 such that for any $\kappa \in \mathscr{I}$ the only non-constant periodic solution of (2.35) that lie in $\mathcal{U}$ are members of the family $p_\epsilon(t)$ for values of ϵ satisfying $\kappa(\epsilon) = \kappa$, $\epsilon \in (0, \epsilon_P)$. The period $T(\epsilon)$ of $p_\epsilon(t)$ is a C^{L+1}-function

$$T(\epsilon) = \frac{2\pi}{\omega_0}\left[1 + \sum_{i=1}^{L/2} \tau_{2i}\epsilon^{2i}\right] + O(\epsilon^{L+1}) \qquad (0 < \epsilon < \epsilon_P).$$

Exactly two of the Floquet exponents of $p_\epsilon(t)$ approach 0 as $\epsilon \to 0$. One is 0 for $\epsilon \in (0, \epsilon_P)$, and the other is a C^{L+1} function

$$\vartheta(\epsilon) = \sum_{i=1}^{L/2} \vartheta_{2i}\epsilon^{2i} + O(\epsilon^{L+1}) \qquad (0 < \epsilon < \epsilon_P).$$

The periodic solution $p_\epsilon(t)$ is orbitally asymptotically stable with asymptotic phase if $\vartheta(\epsilon) < 0$ but is unstable if $\vartheta(\epsilon) > 0$. If there exists a first non-vanishing coefficient κ_{2k} $(1 \leqslant k \leqslant [L/2])$, then there is an $\epsilon_1 \in (0, \epsilon_P]$ such that the open interval

$$\mathscr{I}_1 = \{\kappa : 0 < \kappa/\kappa_{2k} < \kappa(\epsilon_1)/\kappa_{2k}\}$$

has the following properties. For any κ in $\mathscr{I}_1$ there is a unique $\epsilon \in (0, \epsilon_1)$ for which $\kappa(\epsilon) = \kappa$. Hence the family of periodic solutions $p_\epsilon(t)$ $(0 < \epsilon < \epsilon_1)$ may be parametrized as $p(t, \kappa)$ $(\kappa \in \mathscr{I}_1)$. For $\kappa \in \mathscr{I}_1$, the period $T(\kappa)$ and Floquet exponent $\vartheta(\kappa)$ are C^{L+1}-functions of $|\kappa|^{1/k}$. The first non-vanishing coefficient ϑ_{2k} is given by

$$\vartheta_{2k} = -2\alpha'(0)\kappa_{2k}$$

and

$$\operatorname{sgn}\vartheta(\kappa) = \operatorname{sgn}\vartheta_{2k} \qquad (\kappa \in \mathscr{I}_1).$$

Thus the members $p(t, \kappa)$ $(\kappa \in \mathscr{I}_1)$ of the family of periodic solutions are orbitally asymptotically stable with asymptotic phase $\vartheta_{2k} < 0$ and are unstable if $\vartheta_{2k} > 0$.

Proof. The interested reader will find the full proof of the theorem in (HKW81). ◇

The initial approximation of a periodic solution is based on the previous theorem and requires calculation of the terms κ_{2i}, τ_{2i} and ϑ_{2i} for $i = 1, \dots, [L/2]$. The approximation of the periodic solution in the case of non-vanishing first coefficient $\kappa_2 \neq 0$ is good enough with the first triple of coefficients κ_2, τ_2 and ϑ_2 in most of the applications.

The algorithm of evaluation of coefficients is described in (HKW81, Chapter 2) and can be applied to the system that has the $m \leqslant n$ generalized eigenvalues $\Lambda_i(\kappa)$ of the Jacobian matrices at the critical point x^*, κ^*

$$A = \frac{\partial f}{\partial x}(x^*, 0, \kappa^*), \quad \dot{A} = \frac{\partial f}{\partial \dot{x}}(x^*, 0, \kappa^*)$$

and satisfies Theorem 4 requirements (c) and (d):

(a) $\Lambda_1(\kappa^*)$ and $\Lambda_2(\kappa^*)$ are a conjugate pair with vanishing real part,

(b) $\Re\Lambda_1'(\kappa^*) \neq 0$,

(c) $\omega_0 := \Im\Lambda_1(\kappa^*) \neq 0$,

(d) $\Re\Lambda_j(\kappa^*) < 0$, for $j = 3, \dots, m$.

With the proposed algorithm it is possible to make initial approximation of the period as

$$T(\epsilon) = \frac{2\pi}{\omega_0} + O(\epsilon^2),$$

and an initial periodic solution

$$x(\epsilon) = x^* + \epsilon\Re[e^{2\pi it/T}v] + O(\epsilon^2),$$

where v is an eigenvector the corresponds to $\Lambda_1(\kappa^*)$.

For the specified small ϵ, the above presented algorithm allows to find an information about the periodic orbit in the ϵ-neighborhood of the Hopf point. In applications this algorithm will be used in a simplified version for an initial approximation of a periodic solution branching from the non-degenerate Hopf point in Section 3.3.

2.9 Continuation of the Hopf bifurcation point

The numerical continuation of the Hopf point is based on the approach that is used in AUTO2000 (D$^+$02). It uses the imaginary part of the Hopf eigenvalue as an unknown of the continuation problem. The augmented system in Diana for the continuation of the Hopf bifurcation point is given by the following $3n+2$ equations for the variables x, u, w, and ω_0:

$$\left\{ \begin{array}{rcl} f(x,0,\nu) & = & 0, \\ f_{\dot{x}}(x,0,\nu)u\omega_0 + f_x(x,0,\nu)w & = & 0, \\ f_{\dot{x}}(x,0,\nu)w\omega_0 - f_x(x,0,\nu)u & = & 0, \\ ||u||_2^2 + ||w||_2^2 & = & 1, \\ u^T w & = & 0. \end{array} \right. \tag{2.36}$$

In system (2.36) the eigenvector $v_1 = u + iw$ corresponds to the eigenvalue $i\omega_0$ ($\omega_0 > 0$) of the generalized eigenvalues problem

$$f_{\dot{x}}(x,0,\nu)v_1 i\omega_0 + f_x(x,0,\nu)v_1 = 0. \tag{2.37}$$

In the Diana package the solution of the equation system (2.36) is realized in the solver class `HopfPointContinuation`. The class realization is described in Appendix A in Section A.4.3.

The next section presents definitions of degenerate Hopf points and possible extensions for the Hopf point continuation algorithm.

2.10 Singularities of the Hopf bifurcation point

With the parameter continuation of the Hopf bifurcation point it is possible to detect singular points where some conditions of Theorem 4 are violated. At these points new phase portraits of the dynamical system can be produced. In particular, periodic orbits appear generically near single-Hopf points, while homoclinic, quasiperiodic, and chaotic motions exist near double-Hopf points with two pairs of purely imaginary eigenvalues. In the following, possible singularities will be described.

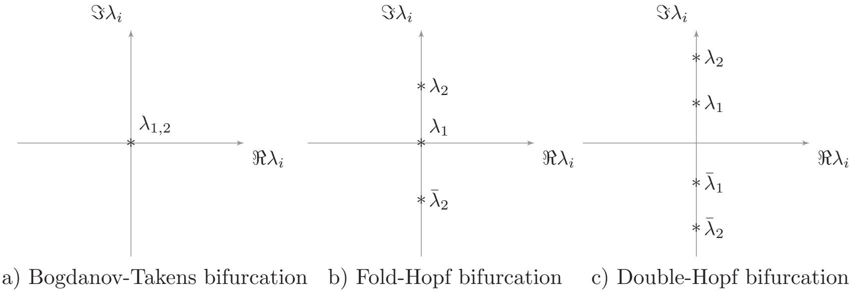


Figure 2.6: Degenerate Hopf Bifurcations

The first kind of singularity appears when ω_0 vanishes and a critical equilibrium has a zero eigenvalue $\lambda_{1,2} = 0$ of algebraic multiplicity 2 but of geometric multiplicity 1 (Figure 2.6a). These are conditions for the *Bogdanov-Takens* (or double-zero) bifurcation. In a two parameter bifurcation diagram at the Bogdanov-Takens point varieties of limit points, Hopf points, and saddle-homoclinic bifurcations coincide. The last bifurcation is global and corresponds to the appearance of *homoclinic orbits* which join a saddle equilibrium point to itself. More precisely, the orbit to an equilibrium at x_0 is called homoclinic if for a point x on the orbit $\varphi^t x \to x_0$ as $t \to \pm\infty$.

The *fold-Hopf* bifurcation in Figure 2.6b appears when a vanishing real eigenvalue coexists with a pair of purely imaginary eigenvalues. The bifurcation point in the parameter plane lies at a tangential intersection of the limit point and the Hopf point varieties. Depending on the system, a branch of torus bifurcations can emanate from the fold-Hopf point. In this case, branches of orbits saddle-focus, homoclinic and *heteroclinic orbits*, occur in the neighborhood of the bifurcation point. The heteroclinic orbit Γ is characterized by the equilibrium points x_1 and x_2, such that for any point x on the orbit Γ the dynamical system attracts to $\varphi^t x \to x_1$ as $t \to -\infty$ and $\varphi^t x \to x_2$ as $t \to +\infty$.

Two pairs of purely imaginary eigenvalues produce a *double-Hopf* or Hopf-Hopf bifurcation (Figure 2.6c) The dynamical system has very rich nonlinear dynamics in a neighborhood of the double-Hopf point. Depending on the particular properties of the system under consideration, there are around thirty different dynamical scenarios, divided into simple and difficult cases. A complex heteroclinic structure is formed in a neighborhood of the bifurcation point. Thus, "strange" dynamics exist near a generic Hopf-Hopf bifurcation. The corresponding parametric portrait has, in addition to local bifurcation curves, a bifurcation set corresponding to global bifurcations, such as, heteroclinic tangencies of equilibrium and cycle invariant manifolds, homoclinic orbits, and associated bifurcations of long-periodic limit cycles.

The final bifurcation with disappearance of the first characteristic exponent ϑ_2 in terms of Theorem 4 is concerned. At this point a *generalized Hopf* bifurcation occurs and a Hopf point variety is separated into two domains with $\vartheta_2 < 0$ and $\vartheta_2 > 0$. For nearby parameter values, the system has two periodic orbits which collide and disappear via a saddle-node bifurcation of periodic orbits. The coexisting stable and unstable periodic orbits can disappear at a non-degenerate fold bifurcation of the periodic orbit bifurcation.

The normal forms derivation for the Bogdanov-Takens, fold-Hopf, double-Hopf, and generalized Hopf bifurcations are described in the book by Y. Kuznetsov (Kuz04, Chapter 8). The book also describes possible phase portraits for the presented bifurcations.

Test functions for the bifurcation described above are presented in (GKS98). The article presents another type of the augmented system for the Hopf point continuation, which does not suffer from the disadvantage, that system (2.36) possesses, which assumes non-zero ω_0. Thus, using (2.36), the solver could not pass a Bogdanov-Takens point where $\omega_0 = 0$ and detect it regularly.

In case the dynamical model has the identity Jacobian matrix $f_{\dot{x}} = I_n$ and can be presented by an ODE system

$$\dot{x} - f(x, \nu) = 0, \tag{2.38}$$

the augmented system can be defined as in the software package CONTENT (KL97). The augmented system, in this specific case, reads

$$\begin{cases} f(x, \nu) &= 0, \\ (f_x^2(x, \nu) + \gamma I_n)u &= 0, \\ \langle u, u \rangle &= 1, \\ \langle L, u \rangle &= 0, \end{cases} \tag{2.39}$$

where $u = \Re v_1$, L is an arbitrary constant vector that is not orthogonal to the kernel of the matrix $f_x^2(x, \nu) + \gamma I_n$.

Obviously, a point with $\gamma > 0$ corresponds to the Hopf point with $\omega_0^2 = \gamma$, for $\gamma < 0$ the point is a neutral saddle, and the critical case with $\gamma = 0$ is the Bogdanov-Takens bifurcation.

2.11 Conclusion

The numerical nonlinear analysis of equilibrium points in continuous dynamic systems has been introduced in this chapter. The main topic of the chapter is the parameter continuation of solutions of the nonlinear algebraic system. Such nonlinear algebraic system can define various phenomena of the dynamical system, for example, the analysis of equilibrium or steady-state points that has been discussed in this chapter. Computation of the parameter dependence of such points is of great interest in engineering applications, because it gives insight in a model behavior with changing parameters.

The base solver, which allows to continuate abstract nonlinear problems with respect to the selected parameter, has been presented. The continuation class implements the predictor-corrector algorithm with the tangent or chord predictors. The corrector is the Newton iteration solver with the local or pseudo-arc length parametrization equation.

As the first solver example, the fixed point continuation solver has been introduced. The solver also computes stability of the equilibrium points with the help of the linearized stability criterion. Critical points where the equilibrium point changes its stability can be detected with the solver via monitoring real parts of the eigenvalues that correspond to the linearization of the differential-algebraic system. Such critical points can be characterized by vanishing real parts of the eigenvalues. In applications, two types of non-degenerate critical points are distinguished.

The first type is the limit or fold point that corresponds to a simple zero real eigenvalue. The limit point and the analysis of singularities of the one-dimensional limit point variety are the second topic of the chapter. At first, the Lyapunov-Schmidt reduction has been presented. The reduction procedure defines a scalar equation that has one-to-one correspondent solutions to the original full system. Also derivatives of the reduced function are derived. Vanishing derivatives of the reduced function determine singular points of the equation solution curve. The singularity theory predicts complex nonlinear behavior in a neighborhood of the most degenerate point. Section 2.5 summarizes mathematical methods of singularity analysis and gives a brief overview of the main results applied to a real bifurcation point variety. The section also presents perturbed bifurcation diagrams for the selected singularity points. Finally, the augmented system and the numerical computation and continuation of the singularity points have been discussed.

The last part of the chapter is dedicated to the Hopf bifurcation, the second type of non-degenerate critical points. The system possesses a periodic orbit branching from the bifurcation point at the bifurcation point. After the C^L-Hopf Bifurcation Theorem statement the algorithm for the computation of the low-order terms in the Poincaré normal form has been presented. The algorithm allows to find an approximation of the periodic orbit in a small neighborhood of the bifurcation point. The augmented system and the solver class for the Hopf point calculation are topics of Section 2.9. Finally a small overview of the degenerate Hopf bifurcation points has been presented.

The implemented nonlinear solvers **SteadyStateContinuation**, **SingAnalyser**, and **HopfPointContinuation**, that are described in Appendix A.4, allow to perform the nonlinear analysis for Diana models. The first solver locates steady-state points of a dynamical system for a given initial state and performs the continuation of such points with respect to the specified parameter. Along the steady state continuation curve the solver can compute generalized eigenvalues of the model linearization. The eigenvalues deliver stability information of the computed steady state. Also the critical points, when the real parts of the eigenvalues change a sign, can be detected.

Critical steady-state points can be used as starting points for the next two solvers. The **SingAnalyser** class makes a parameter continuation of limit points. Additionally, the solver computes derivatives of the reduced function that allow to detect singularities of the solution curve and initiate continuation of a more singular point variety. The derivatives derivatives g_z, g_{zz}, g_λ and $g_{z\lambda}$ of the reduced equation (2.13) have been implemented in the solver for the user-defined parameter $\lambda \in \nu$. With the vanishing derivatives, simple bifurcation hysteresis, pitchfork, and winged cusp points can be found, and the parameter continuation can be used to find varieties of these points.

The last solver **HopfPointContinuation** is used to find a parameter dependence of the Hopf bifurcation point for a dynamical system. The solver can be initiated at a critical steady-state point with a conjugate pair of purely imaginary eigenvalues. The solver only computes a variety of Hopf points without computing the test functions for degenerate or singular points. The implementation of test functions for an ordinary differential equations case has been presented in the last section. Implementation of these functions, which allow to detect degenerate Hopf points, like Bogdanov-Takens, zero-Hopf, double-Hopf, or generalized Hopf bifurcation points, may be used as a part of the future development of the DIANA nonlinear analysis suite.

Chapter 3

Periodic Orbits Continuation and the Recursive Projections Method

Periodic oscillations in chemical engineering studies are widely distributed phenomena and essential for the design and control of processes. Computation of the parameter dependence of periodic oscillations is of great interest in applications, because it gives insight about a model behavior with changing parameters. Numerically such task results a periodic boundary-value problem that can be solved with different well established methods, like shooting or finite difference methods, see (AMR95).

Often a high-dimensional system has only low-dimensional dynamics. The idea of combining Newton's method with a direct solver for a low-dimensional subspace and cheap iterative methods for a higher-dimensional one can be applied for such system. For the computation and analysis of periodic solutions the separation can lead to quite efficient algorithms. The idea was first proposed for steady-state solutions of large symmetric problems by Jarausch and Mackens (JM82; JM84; JM87) — they called their method the "condensed Newton - supported Picard" method — and later extended it to non-symmetric problems by Shroff and Keller (SK93) in their "Recursive Projection Method" or RPM. Further development of the recursive projection method can be found in the comprehensive work by Lust (Lus97; LRSC98), where extensions and improvements of the RPM were analyzed.

More specifically, in this method the property that many chemical engineering models with distributed parameters have periodic orbits with only a few unstable or weakly stable modes are used. The aim of the chapter is to develop a numerical computation of periodic solutions for high-dimensional systems, like, population balance models. The computation of the stability of periodic solutions and the determination of parameter values, at which stability changes occur, will also be presented here.

The chapter is structured as follows: Section 3.1 presents periodic solutions for differential-algebraic equation systems. Numerically such solutions lead to a periodic boundary-value problem with an additional phase condition equation. A notion of the monodromy matrix and characteristic multipliers will also be presented. Stability of periodic solutions in terms of the Poincaré map and non-hyperbolic fixed points of the map in Section 3.2 are presented. The next Section 3.3 summarizes numerical methods for the parameter continuation of periodic solutions and shows initial solution approximation in a neighborhood of the Hopf point in terms of Theorem 4.

The second part of the chapter is dedicated to the recursive projection method. Section 3.4 discusses the method derivation for a fixed value problem of a mapping with a nonlinear algebraic restriction. Also the implemented numerical algorithm in this work is presented. The final Section 3.5 is dedicated to applications of the RPM method to fixed value problems, like, a solution of linear algebraic system, the parameter continuation of steady states and periodic orbits for a dynamical system.

3.1 Periodic solutions

This chapter will mainly be concerned with periodic solutions of an autonomous differential-algebraic equation system in implicit form

$$f(x, \dot{x}, \nu) = 0, \qquad x(0) = x_0, \tag{3.1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is supposed to be differentiable sufficiently many times. In the following let

$$\varphi(x_0, t, \nu), \quad \varphi : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^p \to \mathbb{R}^n \tag{3.2}$$

denote the solution of the equation system (3.1) that is generated by the evaluation operator φ^t and satisfies

$$\begin{aligned} \varphi(x_0, 0, \nu) &= x_0, \\ \varphi(x_0, t+\tau, \nu) &= \varphi(\varphi(x_0, t, \nu), \tau, \nu). \end{aligned} \tag{3.3}$$

A *periodic orbit* or *cycle* is a non-constant closed solution of (3.1) with a constant $T > 0$ such that

$$\varphi(x_0, t, \nu) = \varphi(x_0, t + T, \nu), \quad 0 \leqslant t < T. \tag{3.4}$$

The smallest positive value of T for which this condition holds is called the *period* of the periodic orbit. For the autonomous DAE system (3.1), an orbit is completely determined by one point of the orbit, so it is sufficient to compute only a point x_0 on the periodic orbit. Hence, a periodic orbit can be found as the solution of a boundary value problem for equation (3.1) with a periodic boundary condition

$$\varphi(x_0, T, \nu) = x_0. \tag{3.5}$$

The boundary value problem for the T-periodic differential equations does not fix the solution completely since any solution can be translated freely in time, *i.e.*, if $\varphi(x_0, t, \nu)$ is a solution then so is $\varphi(x_0, t + \tau, \nu)$ for any τ and every point x_0 on the periodic orbit can be used to determine the limit cycle. Therefore, an additional equation

$$s(x_0, T, \nu) = 0 \tag{3.6}$$

that is called a *phase condition* is required. The phase condition selects one periodic solution among infinitely many periodic solutions corresponding to the same periodic orbit but having different initial points. The equation $s(x_0, T, \nu) = 0$ defines a Poincaré surface which is an $(n - 1)$-dimensional affine hyperplane. The orbit should intersect this surface transversely, that is, $\langle \dot{\varphi}, s_{x_0} \rangle \neq 0$ should hold. Possible variants of phase conditions are:

(a) $s(x, T, \nu) := x_i - \bar{x}_i$, for some variable x_i in the vector x_0 and $\bar{x}_i$ is a predicted value in the range of φ_i.

(b) $s(x, T, \nu) := \dot{\varphi}_i(x, 0, \nu)$, that specifies zero time derivative for ith variable at the initial time point.

(c) $s(x, T, \nu) := \dot{\varphi}(x, 0, \nu)^T (x - z)$ is a phase condition with built-in transversality. Here, z is a point near the orbit, for example, a solution on the previous continuation step.

(d) The integral phase condition

$$s(x, T, \nu) := \int_0^T \langle \varphi(x, t, \nu), \dot{\tilde{\varphi}}(t) \rangle \, \mathrm{d}t$$

minimizes the square of L_2-norm

$$D(\tau) := \int_0^T ||\varphi(x, t+\tau, \nu) - \tilde{\varphi}(t)||_2^2 \, \mathrm{d}t$$

of the difference between the solution φ and the previously calculated one $\tilde{\varphi}$ with respect to the time shift τ. This phase condition equation is widely used in the periodic orbits continuation, for example, in AUTO (D+02) or CONTENT (KL97).

With one of the defined phase conditions above the periodic boundary-value problem can be written as a nonlinear-algebraic system for the variables x_0 and T as

$$\begin{aligned} \varphi(x_0, T, \nu) - x_0 &= 0 \\ s(x_0, T, \nu) &= 0 \end{aligned} \tag{3.7}$$

The transversality requirement for the phase condition guarantees that (3.7) will have a nonsingular Jacobian matrix for regular solutions.

The parameter continuation of periodic orbits will be discussed in Section 3.3 and to the rest of the section the parameters vector ν will be fixed.

Another notion that will extensively be used in this chapter is a *monodromy or principal matrix*. The monodromy matrix M for a periodic orbit is defined as a sensitivity matrix with respect to the initial values at a time point T

$$M(x_0, T, \nu) = \left. \frac{\partial \varphi(x_0, t, \nu)}{\partial x_0} \right|_{t=T} \tag{3.8}$$

The monodromy matrix can be obtained from a homogeneous linear variational equation that results after differentiation of $f(\varphi, \dot{\varphi}, \nu) \equiv 0$ with respect to x_0

$$f_x(\varphi, \dot{\varphi}, \nu) \frac{\partial \varphi}{\partial x_0} + f_{\dot{x}}(\varphi, \dot{\varphi}, \nu) \frac{\mathrm{d}}{\mathrm{d}\,t} \frac{\partial \varphi}{\partial x_0} = 0 \tag{3.9}$$

that corresponds to a linear DAE system for the matrix M

$$f_{\dot{x}}(\varphi, \dot{\varphi}, \nu) \dot{M}(x_0, t, \nu) + f_x(\varphi, \dot{\varphi}, \nu) M(x_0, t, \nu) = 0, \quad M(x_0, 0, \nu) = I. \tag{3.10}$$

In terms of the fundamental matrix $\Phi(t)$ for (3.10), the monodromy matrix can be defined as $\Phi(t_0)^{-1}\Phi(t_0+T)$, so the monodromy matrix does not depend on a choice of an arbitrary initial time point t_0. Also the monodromy matrix does not inherit sparsity structure of Jacobian matrices of the underlying DAE system (3.1). In the following, numerical algorithms M will be treated as a full dense matrix $n \times n$.

The matrix-vector product Mv expresses in the first order approximation how a small perturbation at x_0 in the direction v will evaluate in a one cycle. Computation of Mv does not require the full $n \times n$ matrix M. Using the linearity of (3.10), the vector $w := Mv$ can be obtained from an n-dimensional DAE system

$$f_{\dot{x}}(\varphi, \dot{\varphi}, \nu)\dot{w}(t) + f_x(\varphi, \dot{\varphi}, \nu)w(t) = 0, \quad w(0) = v. \tag{3.11}$$

The eigenvalues of M are called the *characteristic multipliers* or *Floquet multipliers* μ. For the periodic solution the monodromy matrix has a unit eigenvalue with the corresponding eigenvector $\dot{\varphi}(x_0, T, \nu)$. In fact, using properties (3.3) and (3.4) the following expression

$$\varphi(x_0, t, \nu)|_{t=T} = \varphi(x_0, t+T, \nu)|_{t=T} = \varphi(\varphi(x_0, t, \nu), T, \nu)|_{t=T}$$

after a differentiation with respect to t at the point $t = T$ gives

$$\dot{\varphi}(x_0, T, \nu) = \frac{\partial \varphi(x_0, T, \nu)}{\partial x_0}\dot{\varphi}(x_0, T, \nu) = M\dot{\varphi}(x_0, T, \nu).$$

Moreover, if μ is a characteristic multiplier of the system (3.10), then a nontrivial solution $\bar{\varphi}$ exists satisfying

$$\bar{\varphi}(x_0, t+T, \nu) = \mu\bar{\varphi}(x_0, t, \nu). \tag{3.12}$$

The proof of this statement can be found in (Far94, p. 53). The vector $\bar{\varphi}(x_0, 0, \nu)$ is an eigenvector corresponding to the multiplier μ. So the characteristic multipliers contain local stability information of the periodic orbit. Namely, the periodic orbit is stable if all of the characteristic multipliers, except trivial one, have a magnitude strictly less than 1; and the periodic orbit is unstable if at least one the characteristic multiplier has a magnitude greater than 1. The non-hyperbolic case with non-trivial characteristic multipliers on the unit circle presents a local bifurcation of the periodic orbit and will be treated in Section 3.2.

It also should be noted that the monodromy matrix for the DAE index-one problem (3.10) may be singular with $\dim\ker M = \dim\ker f_{\dot{x}}$. This fact is based on the existence of a decomposition (LMW98, Theorem 3.1) of the fundamental solution matrix

$$\Phi(t) = H(t) \begin{pmatrix} e^{\vartheta t} & 0 \\ 0 & 0 \end{pmatrix} H(0)^{-1}.$$

where $H \in C^1$ is nonsingular and T-periodic. Exponents ϑ are called *Floquet or characteristic exponents* such that $e^{\vartheta T}$ are characteristic multipliers of the system (3.10).

If the interval $[0, T]$ is split in sub-intervals with m points $0 < t_1 < \ldots < t_m < T$ then the expression

$$\varphi(x_0, T, \nu) = \varphi(\varphi(\ldots \varphi(\varphi(x_0, t_1, \nu), t_2, \nu), t_m, \nu), T, \nu),$$

after the differentiation with respect to x_0, gives

$$M(x_0, T, \nu) = M(x_m, T, \nu) \cdot M(x_{m-1}, t_m, \nu) \cdot \ldots \cdot M(x_1, t_2, \nu) \cdot M(x_0, t_1, \nu), \quad (3.13)$$

where $x_1, \ldots, x_m$ are solution points at $t_1, \ldots, t_m$. Expression (3.13) can be used in the computation of the monodromy matrix in methods with the split time interval, *e.g.,* multiple shooting or collocation methods for the periodic boundary-value problem (3.7).

3.2 Stability and bifurcations of periodic orbits

How the behavior of solutions near a periodic orbit can be investigated in terms of the dynamics of a map will be shown in this section.

The local stability of the periodic orbit can be deduced by analyzing the *Poincaré map* or a *first recurrence map*, which is the intersection of a periodic orbit in the state space of a continuous dynamical system with a certain $(n-1)$-dimensional subspace, called the *Poincaré or transversal section.* The Poincaré section Ω (Figure 3.1) can be given implicitly by a scalar equation $p(x) = 0$ as

$$\Omega = \{x \mid p(x) = 0\} \quad (3.14)$$

an $n - 1$-dimensional hypersurface that cuts the periodic orbit transversely in the first point of a periodic orbit x_0, *i.e.*, $p(x_0) = 0$ and transversality condition $\langle \dot{\varphi}, p_x(x_0) \rangle \neq 0$ is satisfied. The Poincaré map $\Pi : \mathcal{U} \to \Omega$ near a periodic orbit Γ is defined to be the map

$$x \mapsto \varphi(x, T_\Omega(x), \nu) \quad (3.15)$$

for $x \in \mathcal{U}$ where $\mathcal{U}$ is a sufficiently small neighborhood of point x_0 in Ω.

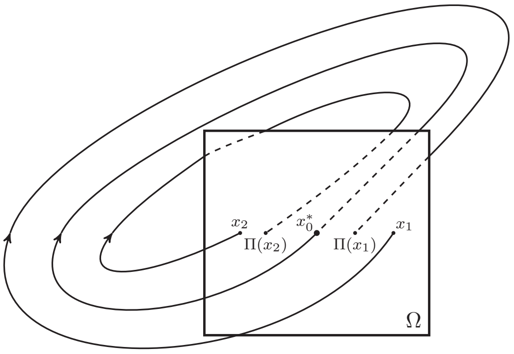


Figure 3.1: A Poincaré section Ω and dynamic system trajectories

The system (3.1) has a periodic solution in a neighborhood of the solution Γ if and only if the Poincaré map has a fixed point, *i.e.*, there is a positive integer k such that $\Pi^k(x_0) = x_0$. The following theorem states correspondence between eigenvalues of the Poincaré map linearization and characteristic multipliers of a variational system (3.9).

Theorem 5 ((Far94, Theorem 5.2.2))**.** *The eigenvalues of the linearization $D\Pi : \Omega \to \Omega$ of the Poincaré map attached to the T-periodic solution Γ of the system* (3.1)*, considered as an $n-1$ dimensional linear operator, are equal to the characteristic multipliers of the variational system with respect to the solution Γ provided that the number 1 is deleted once from the set of the multipliers.*

The local stability of the periodic solution Γ depends on the contractivity of the corresponding Poincaré map and with Theorem 5 on eigenvalues of the monodromy matrix M. If 1 is a simple characteristic multiplier of the variational system (3.9) and the remaining $n-1$ characteristic multipliers are in modulus less then 1, then by the previous theorem all $n-1$ eigenvalues of the Poincaré map linearized at x_0 are in modulus less then 1. This implies that in a neighborhood of x_0 the Poincaré map is a contraction mapping and the periodic orbit is asymptotically stable. The periodic orbit is unstable if at least one of the characteristic multipliers lies outside the unit circle. There are three cases of codimension-1 non-hyperbolic points with the non-trivial unit characteristic multipliers:

- $\mu_i = 1$**: fold bifurcation of a periodic orbit**
  The study of the parameter dependency of periodic solutions can lead to the bifurcations of such solutions when some of the characteristic multipliers leave the unit circle. The first scenario that will be presented here is characterized by a simple eigenvalue 1 of the Poincaré map linearization or a double eigenvalue 1 of the monodromy matrix M with a geometric multiplicity of either one or two. The non-degenerate case of this bifurcation is called a fold point of the periodic orbit. In a sufficiently small neighborhood of the critical parameter value two possible phase portraits can coexist with two periodic solutions (a stable and an unstable one) or without periodic solutions.

- $\mu_i = -1$**: period doubling bifurcation**
  A non-degenerate period doubling occurs when a single characteristic multiplier crosses the unit circle transversally at -1 and no other characteristic multiplier is one in modulus except the trivial characteristic multiplier 1. In a period doubling point, a branch of periodic solutions with approximately doubled period appears or disappears. These solutions go twice around the original periodic orbit. On the period doubled branch, the bifurcation point is seen as a pitchfork bifurcation with two independent eigenvectors. The monodromy matrix at the bifurcation point computed on the period doubled branch is the square of the monodromy matrix on the two-sided branch. The period doubling bifurcation is also known as flip bifurcation or subharmonic bifurcation (the latter since it gives rise to a new solution branch of periodic solutions with approximately half the frequency of the original branch).

- $\mu_i = e^{\pm i\vartheta}$**: bifurcation into torus**
  In this scenario, a pair of complex conjugate eigenvalues of the monodromy matrix crosses the unit circle. Except for this pair and the trivial characteristic multiplier 1, no other characteristic multiplier lies on the unit circle. The torus bifurcation is also known as the Neimark-Sacker bifurcation or Hopf bifurcation of limit cycles.

  In the analysis of the torus bifurcation, there is a distinction between whether the eigenvalues cross the unit circle at a root of unity or not, *i.e.*, between whether there exists an $n \in N : \mu^n = 1$ or not or equivalently, whether $2\pi/\vartheta$ is a rational or irrational number. In the first case, the system has a resonance phenomena. Especially the cases $\vartheta = 2\pi/3$ and $\vartheta = 2\pi/4$ require special analysis. These two regimes are known as strong resonance. If the eigenvalues are another root of unity, the regime is known as a weak resonance. In the latter case, the Poincaré section has at one side of the bifurcation an invariant "circle" of periodic points with alternating stability, corresponding to the existence of two periodic solutions with different stability nearby the original solution. When the critical Floquet multipliers are not a root of unity, there are no new periodic trajectories, but there is an invariant surface at one side of the bifurcation point.

The precise analysis of the phenomena is beyond the scope of this work. A complete analysis can be found in (Arn83; HK91; Kuz04).

3.3 Continuation of periodic solutions

For the continuation of periodic solutions for the differential-algebraic equation system (3.1) the algorithm that has been described in Section 2.2 can be applied. The nonlinear algebraic problem (2.3) is defined for the point x_0 on the periodic orbit Γ, the periodic orbit period T, and a scalar parameter $\kappa \in \nu$. These variables form an $n+2$ dimensional vector $\{x_0, T, \kappa\}$ that is subject to a nonlinear-algebraic problem

$$\left\{ \begin{array}{rcl} \varphi(x_0, T, \kappa) - x_0 & = & 0, \\ s(x_0, T, \kappa) & = & 0, \\ n(x_0, T, \kappa) & = & 0, \end{array} \right. \tag{3.16}$$

where $\varphi(x_0, T, \kappa)$ is a flow generated by the implicit differential-algebraic equation system

$$f(x, \dot{x}, \kappa) = 0, \quad x(0) = x_0, \tag{3.17}$$

$s(x_0, T, \kappa)$ is one of the phase conditions from Section 3.1, and $n(x_0, T, \kappa)$ is the parametrization equation that has been described in Section 2.2.2.

The periodic boundary-value problem can be solved with either shooting or collocation methods (AMR95). In this work only single shooting method with a "black-box" DAE solver is used. The chosen DAE solver is the DASPK (LP00) that, in addition to φ calculation for (3.17), allows to compute the sensitivity of the solution φ with respect to the initial value x_0 and the parameter κ.

The integral phase condition

$$s(x_0, T, \nu) := \int_0^T \langle \varphi(x_0, t, \nu), \dot{\tilde{\varphi}}(t) \rangle \, \mathrm{d}t \tag{3.18}$$

is used in this algorithm and computed also with the DASPK solver. In (3.18) the reference solution $\tilde{\varphi}(t)$ is equal to the previously computed one, or to an initial approximation near the Hopf bifurcation point at the first continuation point. The DAE system (3.17) is extended with an additional variable s and an equation

$$\dot{s} = \langle x, \dot{\tilde{\varphi}}(t) \rangle, \quad s(0) = 0,$$

where $\dot{\tilde{\varphi}}(t)$ is a time derivative of a smooth interpolation of the reference solution. This approach allows to use DASPK to compute derivatives $\partial s / \partial x_0$ and $\partial s / \partial \kappa$.

The parametrization equation $n(x_0, T, \kappa)$ in (3.16) does not differ from the one, described in Section 2.2.2, and may be either local (2.6) or pseudo-arc length (2.7).

The Newton update Jacobian matrix for the corrector step of the continuation algorithm 2.2.4 can be formulated for the boundary-value problem (3.16) described above as

$$J(x_0, T, \kappa) := \left[\begin{array}{ccc} \varphi_{x_0}(x_0, t, \kappa) - I & \dot{\varphi}(x_0, t, \kappa) & \varphi_\kappa(x_0, t, \kappa) \\ s_{x_0}(x_0, t, \kappa) & \dot{s}(x_0, t, \kappa) & s_\kappa(x_0, t, \kappa) \\ n_{x_0}(x_0, t, \kappa) & n_T(x_0, t, \kappa) & n_\kappa(x_0, t, \kappa) \end{array} \right]_{t=T}, \tag{3.19}$$

which is a dense $(n+2) \times (n+2)$ matrix. It also can be proven that J is nonsingular for regular points of the periodic solution branch. Computation of φ_{x_0} and s_{x_0} requires a solution of the linear variational system for (3.17) and (3.18)

$$\begin{array}{rcl} f_{\dot{x}}(\varphi, \dot{\varphi}, \kappa)\dot{\varphi}_{x_0}(t) + f_x(\varphi, \dot{\varphi}, \kappa)\varphi_{x_0}(t) & = & 0 \\ \dot{s}_{x_0}(t) - \langle \varphi_{x_0}(t), \dot{\tilde{\varphi}}(t) \rangle & = & 0 \end{array}, \quad \text{with} \quad \begin{array}{rcl} \varphi_{x_0}(0) & = & I \\ s_{x_0}(0) & = & 0 \end{array}, \tag{3.20}$$

and for the φ_κ and s_κ

$$\begin{array}{rcl} f_{\dot{x}}(\varphi, \dot{\varphi}, \kappa)\dot{\varphi}_\kappa(t) + f_x(\varphi, \dot{\varphi}, \kappa)\varphi_\kappa(t) + f_\kappa(\varphi, \dot{\varphi}, \kappa) & = & 0 \\ \dot{s}_\kappa(t) - \langle \varphi_\kappa(t), \dot{\tilde{\varphi}}(t) \rangle & = & 0 \end{array}, \quad \text{with} \quad \begin{array}{rcl} \varphi_\kappa(0) & = & 0 \\ s_\kappa(0) & = & 0 \end{array}. \tag{3.21}$$

With the DASPK solver the initial value problem (3.17) and the linearized variational equation systems (3.20)–(3.21) are solved jointly for the time interval $[0, T]$. The computed Newton update matrix J is used to produce the following iteration process

$$\left[\begin{array}{c} x_0^{(i+1)} \\ T^{(i+1)} \\ \kappa^{(i+1)} \end{array} \right] = \left[\begin{array}{c} x_0^{(i)} \\ T^{(i)} \\ \kappa^{(i)} \end{array} \right] - J^{-1} \left[\begin{array}{c} \varphi(x_0^{(i)}, T^{(i)}, \kappa^{(i)}) - x_0^{(i)} \\ s(x_0^{(i)}, T^{(i)}, \kappa^{(i)}) \\ n(x_0^{(i)}, T^{(i)}, \kappa^{(i)}) \end{array} \right], \tag{3.22}$$

for the corrector procedure described in 2.2.2. The presented method is applicable only for small enough systems because an n-dimensional model requires computation of $(n+1)^2$ sensitivity variables in terms of the variational systems (3.20)–(3.21).

Another issue that will be covered in this section is the initial approximation of the periodic solution. In this work an approach based on the algorithm that has been described in Section 2.8 is used. In a neighborhood of the Hopf point the period T can be approximated by $2\pi/\omega_0$, where ω_0 is a positive imaginary part of the critical eigenvalue. The initial periodic solution is approximated with

$$\begin{aligned}\tilde{\varphi}(t) &= x^* + \epsilon e^{2\pi i t/T} v_1 = x^* + \epsilon(\cos(2\pi t/T)\Re v_1 - \sin(2\pi t/T)\Im v_1),\\ \dot{\tilde{\varphi}}(t) &= \frac{2\pi\epsilon}{T} e^{2\pi i t/T} v_1 = \frac{2\pi\epsilon}{T}(-\sin(2\pi t/T)\Re v_1 - \cos(2\pi t/T)\Im v_1),\end{aligned} \tag{3.23}$$

where x^* is the Hopf bifurcation point, v_1 is an eigenvector corresponding to the critical eigenvalue $i\omega_0$, and ϵ defines a small neighborhood near the Hopf point.

3.4 The recursive projection method

The recursive projection method (SK93; Lus97) addresses the problem that finds a fixed point x^* of a recursive iteration

$$x^{(i+1)} = F(x^{(i)}, \nu) \tag{3.24}$$

such that

$$x^* = F(x^*, \nu), \tag{3.25}$$

for the smooth enough mapping $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$.

The usual way to find the fixed point x^* in (3.24) is the fixed point iteration that generates a sequence of the points $x_0, x_1, x_2, \ldots$ which are hoped to converge to the point x^*. The initial point x_0 is supposed to be in a sufficient neighborhood of the solution point x^*. The convergence speed and the stability of fixed point iteration are determined by eigenvalues of the map linearization F_x at the point x^*. The method is divergent if at least one eigenvalue lies outside the unit disk, and convergent if all eigenvalues are inside the unit disk. But the method is convergent with slow convergence speed if some eigenvalues lie near the unit disk. The other way to find the fixed point x^* is to use the Newton iteration for the nonlinear-algebraic problem

$$F(x, \nu) - x = 0.$$

But this method requires computation of the full matrix F_x on each iteration that requires considerable numerical cost for $n \gg 1$. In this section the RPM algorithm that combines quadratic convergence of the Newton method for a low-dimensional solution subspace and the low-computational cost fixed point Picard iteration for a high-dimensional subspace will be presented.

Also in different applications to (3.24) a nonlinear algebraic constraint G may be added that satisfy in implicit form

$$G(x, \nu) = 0.$$

The nonlinear constraint G should satisfy requirements of the Implicit Function Theorem to make a parametrization of the solution $x^*(\nu)$. Such constraints may be a parametrization equation for the parameter continuation of the solution x^* or a phase condition in the periodic solutions computation for fixing of the solution with respect to a time shift. This two examples of additional nonlinear equations will be discussed below in Section 3.5.

3.4.1 Mathematical background

The recursive projection method exploits the fact that slow convergence speed or divergence of (3.24) is caused by a small number of eigenvalues that have modulus greater than $1 - \varrho$ for a user-defined positive value ϱ. The idea of the method is to split the solution space into a “stable” or fast converging subspace and an orthogonal “unstable” one, where the solution has slow convergence or diverges. The splitting allows to apply different numerical algorithms in each subspace. In this work, the Picard iteration in the “stable” subspace and the Newton iteration in the “unstable” one will be used. Combination of this methods results in the so-called Newton-Picard method that was introduced in a work by Shroff and Keller (SK93) and extended in a work by Lust (Lus97).

Let the task be defined in terms of (3.24) as

$$\begin{array}{lcll} x^{(i+1)} & = & F(x^{(i)}, \nu), & x \in \mathbb{R}^n, \\ 0 & = & G(x^{(i)}, \nu), & \nu \in \mathbb{R}^m, \end{array} \tag{3.26}$$

where $x^{(i)} \in \mathbb{R}^n$ is a vector of unknown variables for an iteration i, $\nu \in \mathbb{R}^m$ is a parameters vector, $F : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is a sufficiently smooth map and $G : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^p$ are smooth nonlinear constraints that are used to find a parametrization of x

with respect to ν The objective of the task (3.26) is to find such a point $\{x^*, \nu^*\}$ that fulfill the following equalities

$$\begin{aligned} x^* &= F(x^*, \nu^*), \\ 0 &= G(x^*, \nu^*). \end{aligned} \tag{3.27}$$

The Newton iteration can be used to solve the task mentioned above in a nonlinear-algebraic form

$$\begin{aligned} F(x, \nu) - x &= 0, \\ G(x, \nu) &= 0. \end{aligned} \tag{3.28}$$

The Newton update on ith iteration is computed from the following linear algebra problem

$$\begin{bmatrix} F_x^{(i)} - I & F_\nu^{(i)} \\ G_x^{(i)} & G_\nu^{(i)} \end{bmatrix} \begin{bmatrix} \Delta x^{(i)} \\ \Delta \nu^{(i)} \end{bmatrix} = - \begin{bmatrix} r^{(i)} \\ G^{(i)} \end{bmatrix} \tag{3.29}$$

where $r^{(i)} := F(x^{(i)}, \nu^{(i)}) - x^{(i)}$ is a residual vector. With the computed values $\Delta x^{(i)}$ and $\Delta \nu^{(i)}$ the iteration update is $x^{(i+1)} = x^{(i)} + \Delta x^{(i)}$, $\nu^{(i+1)} = \nu^{(i)} + \Delta \nu^{(i)}$. This numerical approach is very expensive in many cases, when the calculation of the first order derivative matrix $F_x^{(i)}$ is the time-consuming part.

To avoid complete computation of the matrix $F_x^{(i)}$ a stabilization procedure is proposed (SK93). Let $F^* := F_x(x^*, \nu^*)$ have eigenvalues $\{\mu_k\}_1^n$ that for some $\varrho > 0$ are ordered as

$$|\mu_1| \geqslant \cdots \geqslant |\mu_{n_p}| > 1 - \varrho \geqslant |\mu_{n_p+1}| \geqslant \cdots \geqslant |\mu_n| \tag{3.30}$$

and $n_p \ll n$. Then it is possible to define the maximal invariant subspace $\mathcal{U}$ of F^* belonging to $\{\mu_k\}_1^{n_p}$ eigenvalues with projectors P and $Q := I - P$ that induce an orthogonal direct sum decomposition

$$\mathbb{R}^n = \mathcal{U} \oplus \mathcal{U}^\perp = P\mathbb{R}^n \oplus Q\mathbb{R}^n.$$

A subspace decomposition can be introduced for the mapping F in (3.28)

$$\begin{aligned} 0 &\overset{!}{=} Q(F(x, \nu) - x) &= QF(p + q, \nu) - q, \qquad & q \equiv Qx \in \mathcal{U}^\perp, \\ 0 &\overset{!}{=} P(F(x, \nu) - x) &= PF(p + q, \nu) - p, \qquad & p \equiv Px \in \mathcal{U}, \end{aligned} \tag{3.31}$$

so the linearization of the first mapping in $\mathcal{U}^\perp$ has eigenvalues $|\lambda(QF^*Q)| \leqslant 1 - \varrho$ with a magnitude less than one and is stable for the fixed point iteration in $\mathcal{U}^\perp$

subspace.

Let $V_p \in \mathbb{R}^{n\times n_p}$ define an orthonormal basis for $\mathcal{U}$ that is spanned by the eigenvectors of F^* corresponding to $\{\mu_k\}_1^{n_p}$ and V_q is an orthonormal basis for $\mathcal{U}^\perp$ with $V_q \perp V_p$, so

$$\begin{array}{rclcl} P & = & V_pV_p^T, & & \bar{p} = V_p^T x \in \mathbb{R}^{n_p}, \qquad p = V_p\bar{p}, \\ Q & = & V_qV_q^T & = \ I - V_pV_p^T, & \bar{q} = V_q^T x \in \mathbb{R}^{n-n_p}, \quad q = V_q\bar{q}, \end{array}$$

where $\bar{p}$ and $\bar{q}$ are solution coordinates in “unstable” and “stable” subspaces. With the projectors P and Q the original task (3.29) can be rewritten as

$$\begin{array}{rclclcl} \bar{r}_q & = & V_q^T r & = & V_q^T(QF(V_p\bar{p} + V_q\bar{q}, \nu) - V_q\bar{q}) & = & 0 \\ \bar{r}_p & = & V_p^T r & = & V_p^T(PF(V_p\bar{p} + V_q\bar{q}, \nu) - V_p\bar{p}) & = & 0 \\ & & & & G(V_p\bar{p} + V_q\bar{q}, \nu) & = & 0 \end{array}$$

and the Newton iteration is transformed to

$$\begin{bmatrix} V_q^T(F_x^{(i)} - I)V_q & V_q^T(F_x^{(i)} - I)V_p & V_q^TF_\nu^{(i)} \\ V_p^T(F_x^{(i)} - I)V_q & V_p^T(F_x^{(i)} - I)V_p & V_p^TF_\nu^{(i)} \\ G_x^{(i)}V_q & G_x^{(i)}V_p & G_\nu^{(i)} \end{bmatrix} \begin{bmatrix} \Delta\bar{q}^{(i)} \\ \Delta\bar{p}^{(i)} \\ \Delta\nu^{(i)} \end{bmatrix} = - \begin{bmatrix} V_q^Tr^{(i)} \\ V_p^Tr^{(i)} \\ G^{(i)} \end{bmatrix}.$$

Using the fact that $V_p^TV_p = I_p$, $V_q^TV_q = I_q$ and $V_p \perp V_q$, the previous system can be separated

$$\begin{bmatrix} V_q^TF_x^{(i)}V_q - I_q & 0 & 0 \\ 0 & V_p^TF_x^{(i)}V_p - I_p & V_p^TF_\nu^{(i)} \\ 0 & G_x^{(i)}V_p & G_\nu^{(i)} \end{bmatrix} \begin{bmatrix} \Delta\bar{q}^{(i)} \\ \Delta\bar{p}^{(i)} \\ \Delta\nu^{(i)} \end{bmatrix} = - \begin{bmatrix} V_q^T(r^{(i)} + F_\nu^{(i)}\Delta\nu^{(i)}) \\ V_p^T(r^{(i)} + F_x^{(i)}V_q\Delta\bar{q}^{(i)}) \\ G^{(i)} + G_x^{(i)}V_q\Delta\bar{q}^{(i)} \end{bmatrix} \tag{3.32}$$

The resulting system (3.32) has the following properties:

(a) The spectrum of the restricted to $\mathcal{U}^\perp$ matrix $F_x^{(i)}$ has eigenvalues that satisfy $|\lambda(V_q^TF_x^{(i)}V_q)| \leqslant 1 - \varrho$. This gives a possibility to apply matrix-free iterative schemes to calculate the solution in $\mathcal{U}^\perp$. The iterative scheme in this work is the Picard iteration, but it is possible to apply other schemes, for example, Krylov subspace methods BiCGSTAB or GMRES (GV96; Saa00).

(b) relatively small size of the bottom right part $(n_p + m) \times (n_p + m)$, so the Newton update can be applied in $\mathcal{U}$ space with low computational cost.

3.4.2 Numerical algorithm

In this section the RPM algorithm for the setting up (3.32) and its solution is presented. The interested reader will find the detailed analysis of the method in a work by Lust (Lus97).

The recursive projection method substitutes the corrector step in the parameter continuation algorithm that has been described in Section 2.2.4. The Newton-Picard corrector algorithm schematically can be presented

```
Input: $x^{(0)}$, $\nu^{(0)}$, $V_p^{(0)}$
Output: $x^*$, $\nu^*$, $\mu$
begin
    $i \leftarrow 0$
    repeat
        $V_p^{(i)}, \mu^{(i)} \leftarrow$ power_iteration($V_p^{(i)}, x^{(i)}, \nu^{(i)}$)
        $\Delta\bar{q}_r^{(i)}, \Delta\bar{q}_\nu^{(i)} \leftarrow$ picard_iteration($V_p^{(i)}, x^{(i)}, \nu^{(i)}$)
        $\Delta\bar{p}^{(i)}, \Delta\nu^{(i)} \leftarrow$ newton_iteration($V_p^{(i)}, x^{(i)}, \nu^{(i)}, \Delta\bar{q}^{(i)}$)
        $\Delta x^{(i)} \leftarrow V_p^{(i)}\Delta\bar{p}^{(i)} + \Delta q_r^{(i)} + \Delta q_\nu^{(i)}\Delta\nu^{(i)}$
        if $||\{r^{(i)},\ G^{(i)}\}||_2 < \epsilon_f$ or $||\{\Delta x^{(i)},\ \Delta\nu^{(i)}\}||_2 < \epsilon_x$ then
            break
        end
        $x^{(i+1)} \leftarrow x^{(i)} + \Delta x^{(i)}$
        $\nu^{(i+1)} \leftarrow \nu^{(i)} + \Delta\nu^{(i)}$
        $V_p^{(i+1)} \leftarrow$ basis_update($V_p^{(i)}, \mu^{(i)}$)
        $i \leftarrow i+1$
    until $i < k_{max}$ ;
    $x^* \leftarrow x^{(i)}$
    $\nu^* \leftarrow \nu^{(i)}$
    $\mu \leftarrow \mu^{(i)}$
end
```

Algorithm 2: Newton-Picard corrector step

The solution of the separated system (3.32) can be done in the following stages of Algorithm 2:

(a) The first step of the algorithm (the function `power_iteration` in Algorithm 2) is the computation of the orthonormal basis V_p for $\mathcal{U}$ and a restriction of the matrix F_x to the subspace $\mathcal{U}$. The algorithm does not require computation of the full matrix F_x, but only a matrix-matrix product $W := F_x V_p$, where V_p has size $n \times n_p$. The matrix restriction is $V_p^T F_x V_p = V_p^T W$ and contains first

n_p ordered by magnitude eigenvalues of F_x. This step is performed with the power subspace iteration (Saa92, Chapter V).

```
Input: $V_p^{[0]}$
Output: $V_p$ with $\mathcal{U} \approx \text{span}\{V_p\}$, $\mu$
begin
    $W = V_p^{[0]}$
    repeat
        $V_p \leftarrow \texttt{orth}(W)$; $W \leftarrow F_x^{(i)} V_p$
        $[S, Y] \leftarrow \texttt{schur}(V_p^T W)$
        descending reordering of $S$ and $Y$ to satisfy (3.30)
        $\mu \leftarrow \texttt{diag}(S)$
        $V_p \leftarrow V_p Y$; $W \leftarrow WY$
    until convergence of $\mu(1 : n_p)$ ;
    $V_p \leftarrow V_p(1 : n_p)$
end
```

Algorithm 3: Power subspace iteration

The linear algebra functions are based on the LAPACK library. The function `orth`(W) computes an orthonormal basis for the range of W and is based on the QR-decomposition (`orgqr` LAPACK function). The function `schur`(A) produces a quasitriangular Schur matrix S and a unitary matrix Y so that $A = YSY^T$ and $Y^TY = I$. This function uses the `gees` LAPACK function. After computing S and Y, the matrices are reordered accordingly to (3.30).

The method for the basis computation is not restricted to the power subspace iteration. Alternatives for this method may be other matrix-free methods for eigenproblems, for example, based on the idea of Krylov subspaces (GV96; Saa92). Here may be mentioned Lanczos and Arnoldi iterations for symmetric and possibly non-symmetric matrices, that are realized in the ARPACK library (LSY98).

(b) In the second step (the function `picard_iteration` in Algorithm 2) the solution update $\Delta\bar{q}$ with an iterative method in Q subspace is computed by

$$\left[V_q^T(F_x - I)V_q\right]\Delta\bar{q} = -V_q^T(r + F_\nu\Delta\nu).$$

With the property $|\lambda(V_q^T F_x V_q)| < 1$ the Picard iteration

$$\Delta\bar{q}^{(k+1)} = V_q^T F_x V_q \Delta\bar{q}^{(k)} + V_q^T(r + F_\nu\Delta\nu), \quad k = 1, 2, \ldots, n_{pic},$$

is convergent. Reformulating the iteration to the equivalent Neumann series the update $\Delta q^{(k)}$ is read for n_{pic} iterations

$$\begin{aligned} V_q \Delta \bar{q} &= \Delta q = (I - V_p V_p^T) \sum_{k=1}^{n_{pic}} F_x^k \cdot (r + F_\nu \Delta \nu) \\ &:= \Delta q_r + \Delta q_\nu \Delta \nu, \end{aligned} \tag{3.33}$$

where Δq_r and Δq_ν are the vector x updates in the subspace $\mathcal{U}^\perp$ that correspond to the residual vector r and parameters ν respectively.

(c) In the third step (the function `newton_iteration` in Algorithm 2) the solution in the P subspace with a direct linear algebra method (the LAPACK function `gesv`) is implemented for a system

$$\begin{bmatrix} V_p^T(F_x - I)V_p & V_p^T(F_\nu + F_x \Delta q_\nu) \\ G_x V_p & G_\nu + G_x \Delta q_\nu \end{bmatrix} \begin{bmatrix} \Delta \bar{p} \\ \Delta \nu \end{bmatrix} = - \begin{bmatrix} V_p^T(r + F_x \Delta q_r) \\ G + G_x \Delta q_r \end{bmatrix}.$$

(d) The solution x is updated with a vector

$$\Delta x^{(i)} = V_p \Delta \bar{p} + \Delta q_r + \Delta q_\nu \Delta \nu,$$

so the next solution iteration will be

$$\begin{aligned} x^{(i+1)} &= x^{(i)} + \Delta x^{(i)} \\ \nu^{(i+1)} &= \nu^{(i)} + \Delta \nu^{(i)}. \end{aligned}$$

(e) Exit conditions for the Newton-Picard iterations in this work are

(1) for $||\{r^{(i)},\ G^{(i)}\}||_2 < \epsilon_f$ or $||\{\Delta x^{(i)},\ \Delta \nu^{(i)}\}||_2 < \epsilon_x$ the current solution is accepted as a fixed point,

(2) if the maximum number of iterations is exceeded, the recursive projection method fails.

(f) As the recursive projection method progresses, the basis V_p for $\mathcal{U}$ will change. A basis update (the function `newton_iteration` in Algorithm 2) should be performed after each iteration if the number of eigenvalues $\mu(1 : n_p)$ is changed during the iteration. The basis vectors $V_p^{(i+1)}$ for the next iteration contain only eigenvectors of F_x that correspond to the eigenvalues $|\mu| > 1 - \varrho$.

In the following section some examples of recursive projection method applications will be presented.

3.5 Application of the recursive projection method

The RPM algorithm described above can be applied to different numerical problems to speedup the convergence rate of the iterations method or even make it convergent. Such problems may be iterative solution of the linear algebra systems, or continuation of steady-state and periodic solutions for dynamical systems. In this section these three numerical problems are presented in form (3.26). The problems also are supplied with simple examples to give an insight into the method.

3.5.1 Linear algebraic problem

For the problem $Ax = b$ with $x, b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ the Picard fixed point iteration reads

$$x^{(i+1)} = (I - A)x^{(i)} + b,$$

and convergence of the iteration process depends on the eigenvalues of the matrix $I - A$. The task in form (3.26) for the recursive projection method can be presented as

$$\begin{aligned} F(x) &:= (I - A)x + b, \quad & x^{(0)} = 0 \in \mathbb{R}^n, \\ G(x) &:= \emptyset, & \nu \in \emptyset. \end{aligned} \tag{3.34}$$

Application of the RPM to (3.34) results the solution vector x^* such that $Ax^* = b$, and the dominant n_p eigenvalues μ of the matrix $I - A$ with the corresponding eigenvectors V_p that belong to the low-dimensional "unstable" subspace.

The following example shows an application of the RPM to a linear algebraic problem for the matrix A that has the size $n = 100$. In the example the matrix $I - A$ has 5 eigenvalues outside the unit disk (Figure 3.2). The results of the system solution with the RPM are shown in Figure 3.3. Solid lines are residual norms $||x^{(i)} - x^*||_2$ of the stabilized process, dashed lines are residual norms of the fixed point iterations without the RPM stabilization. Points with $\circ$ marks denote Newton updates after every five Picard iterations. After every Newton update the number of "unstable" eigenvalues and the dimension of the subspace $\mathcal{U}$ is compared. In case if the number of such eigenvalues is more than the subspace dimension $\dim \mathcal{U}^{(i)}$, the basis $V_p^{(i+1)}$ for the next Picard iteration is extended by one vector. The only difference in sub-figures of Figure 3.3 is the initial approximation

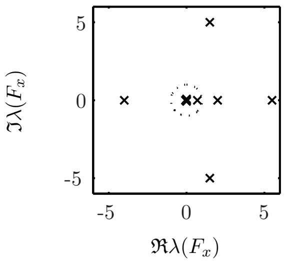
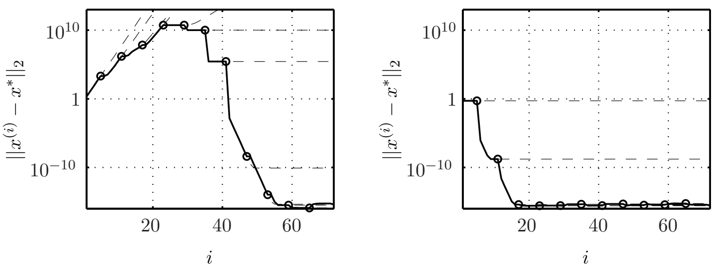


Figure 3.2: Eigenvalues of the test matrix $F_x := (I - A)$



a) empty initial subspace $\mathcal{U} = \emptyset$ b) initial subspace with dim $\mathcal{U}^{(0)} = 5$

Figure 3.3: Residual norms for Picard iterations; dashed lines are residual norms without the RPM stabilization; the RPM stabilization is performed after every five Picard iterations (denoted by o marks); at this points the solution $x^{(i)}$ is corrected by a Newton update and the basis $V_p^{(i)}$ is updated

of the basis $V_p^{(0)}$. In the left sub-figure the algorithm starts with an empty initial "unstable" subspace $\mathcal{U}$ and in the right sub-figure it starts with the subspace $\mathcal{U}$ spanned by five randomly generated vectors. This leads to a difference in the convergence of the iteration process. The iterative process in the first example after only five basis updates is stabilized ($i = 30$) and afterward it is similar to the second one.

This example shows significance of the initial "unstable" subspace $\mathcal{U}$ and of the basis update strategy. Namely, starting with an empty $\mathcal{U}$ or extension of the basis $V_p^{(i)}$ with a small number of eigenvectors may lead to slow convergence (60 iterations in the first case and 18 in the second one).

3.5.2 Calculation of steady-state solutions

The recursive projection method can be applied for a continuation of an equilibrium point of an autonomous DAE system

$$f(x, \dot{x}, \gamma) = 0, \quad f : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$$

with a dynamic flow $\varphi(x_0, t, \gamma)$, such that $f(\varphi, \dot{\varphi}, \gamma) \equiv 0$ and $\varphi(x_0, 0, \gamma) = x_0$. The equilibrium point in terms of a phase flow φ can be defined as a fixed point

$$x^* = \varphi(x^*, \tau, \gamma)$$

for some time interval $[0, \tau]$ with $\tau > 0$. The task in form (3.26) can be presented as

$$\begin{aligned} F(x, \nu) &:= \varphi(x, \tau, \gamma), \qquad & x^{(0)} &= x_0 \in \mathbb{R}^n, \\ G(x, \nu) &:= n(x, \gamma), & \nu &= \{\gamma\} \in \mathbb{R}, \end{aligned} \tag{3.35}$$

where $n(x, \gamma)$ is the parametrization equation that has been defined in 2.2.2. The Newton-Picard method substitutes the corrector step in the predictor-corrector algorithm from Section 2.2.4. The corrector applied to (3.35) results the steady-state solution point x^*, γ^* such that $x^* = \varphi(x^*, \tau, \gamma^*)$. As a byproduct the algorithm computes the dominant n_p eigenvalues μ of the fundamental matrix $\Phi(\tau)$ of homogeneous linear variational equation (3.9). An approximation of the basis for the "unstable" subspace $\mathcal{U}$ is also computed and defined by V_p basis vectors. The dominant eigenvalues μ contain stability information of the steady-state point: if at least one eigenvalue lies outside the unit disk the equilibrium point is unstable, and is stable if all eigenvalue's magnitudes are less then one.

3.5.3 Calculation of periodic solutions

A natural approach to compute stable periodic solutions is the straightforward time integration of (3.1). The time integration for a boundary-value problem (3.16) is equivalent to a fixed point iteration

$$x^{(i+1)} = \varphi(x^{(i)}, T(x^{(i)}), \nu), \tag{3.36}$$

which can be a subject of the recursive projection method that has been described above.

The periodic solution continuation system in form (3.26) can be presented as

$$\begin{array}{ll} F(x,\nu) := \varphi(x,T,\nu) & x^{(0)} = x_0 \in \mathbb{R}^n, \\ G(x,\nu) := \left\{ \begin{array}{c} s(x,T,\nu) \\ n(x,T,\nu) \end{array} \right\} & \nu = \{T,\gamma\} \in \mathbb{R}^2, \end{array}$$

where $n(x,T,\nu)$ is a parametrization equation (Section 2.2.2) and $s(x,T,\nu)$ is a phase condition that has been defined in Section 3.1. For this task the Newton-Picard method also substitutes the corrector step in the predictor-corrector algorithm from Section 2.2.4. The Newton-Picard corrector results the corrected periodic solution point x^*, γ^* with period T^* such that

$$\begin{array}{rcl} x^* &=& \varphi(x^*,T^*,\gamma^*), \\ 0 &=& s(x^*,T^*,\gamma^*), \\ 0 &=& n(x^*,T^*,\gamma^*). \end{array} \tag{3.37}$$

In addition it computes the dominant n_p eigenvalues μ of the monodromy matrix M. An approximation of the basis for the "unstable" subspace $\mathcal{U}$ is also computed and defined by V_p basis vectors. The dominant eigenvalues μ of the matrix $V_pF^*V_p^T$ are characteristic multipliers and contain stability information of the periodic orbit.

In the following subsection, a simple example gives an insight in the recursive projection method that is applied to a two-dimensional dynamical model.

3.5.4 Show case example

The steady state and periodic solution continuation is shown by a two-dimensional dynamical system defined by

$$\begin{array}{rcll} \dot{x}_1 &=& 2\pi x_2 + \delta(R-(x_1^2+x_2^2))x_1, & x_1(0) = x_{1,0}, \\ \dot{x}_2 &=& -2\pi x_1 + \delta(R-(x_1^2+x_2^2))x_2, & x_2(0) = x_{2,0}. \end{array}$$

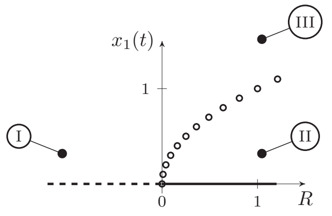
The system can be transformed in polar coordinates as follows

$$\begin{array}{rcll} \dot{\rho} &=& \delta\rho(R-\rho^2), & \rho(0) = \rho_0 > 0, \\ \dot{\theta} &=& 2\pi, & \theta(0) = \theta_0, \end{array}$$

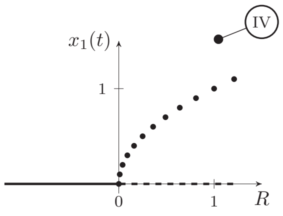
and solved for $\rho(t,\rho_0)$, $\theta(t,\theta_0)$

$$\begin{aligned}\rho(t,\rho_0) &= \pm\frac{e^{\delta\,Rt}\sqrt{R}\,\rho_0}{\sqrt{R\,-\,\rho_0^2(1-e^{2\,\delta\,Rt})}},\\ \dot{\theta} &= 2\pi t+\theta_0\end{aligned}$$

A steady-state solution of the system with radius 0 (ρ=0) exists for all R, see continuation diagrams in Figure 3.4. Stability of the steady-state is determined by a product δR, for a positive value of the product the steady-state is unstable, and for negative is stable. However, when R is positive, there also coexists a periodic solution of fixed radius $\sqrt{R}$ with a period 1. Stability of the periodic branch also determined by the parameter δ, for a positive value the branch is stable and for negative is unstable. The limiting case at the origin is a Hopf bifurcation. Depending on the sign of the parameter δ the bifurcation point may be either a subcritical Hopf bifurcation (Figure 3.4a) or a supercritical Hopf bifurcation (Figure 3.4b). In the figure, solid and dashed lines present stable and unstable



a) continuation diagram for $\delta < 0$



b) continuation diagram for $\delta > 0$

Figure 3.4: Continuation of steady states and periodic solutions; solid and dashed lines are stable and unstable steady-state branches; filled and empty circles are stable and unstable periodic solution branches; points show initial conditions for stabilization examples in Figures 3.5 – 3.6

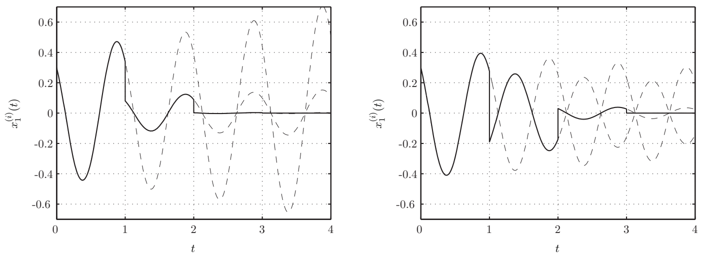
steady state branches. Periodic solution branches are indicated with circles. Filled and empty circles branches correspond respectively to stable and unstable periodic solutions.

To show the stabilization process at four different cases I – IV in Figure 3.4 the recursive projection algorithm is applied. The initial values and parameters are presented in Table 3.1. Results of the Newton-Picard stabilization for different parameters and initial values are shown in Figures 3.5 – 3.6. The dimension of

Point	Figure	$x_{1,0}^{(0)}$	$x_{2,0}^{(0)}$	R	δ
I	3.5a	0.3	0.3	-1	-0.1
II	3.5b	0.3	0.3	1	-0.1
III	3.6a	1.5	0.0	1	-0.1
IV	3.6b	1.5	0.0	1	0.1

Table 3.1: The initial values and parameters for the test examples I – IV

the unstable subspace is constant and the unstable subspace basis consists of two vectors. In both cases the parametrization equation fixes the parameter value R (Table 3.1). For periodic solution branches an additional phase condition fixes the period $T = 1$. The stabilization procedure of steady states periodic solution branches is applied with a fixed time interval $\tau = 1$. Solid lines in Figures 3.5 – 3.6 show the iterative stabilization process with the stabilization Newton steps at time points $t = 1, 2, \ldots$ and dashed lines show dynamic simulations that start at time points $t = 0, 1, 2 \ldots$ without the stabilization. The stabilization iteration in the example can be treated as one Picard step for a map $x^{(i+1)} := \varphi(x^{(i)}, 1, R)$ with a consequent Newton step. Computed fixed points and corresponding eigenvalues

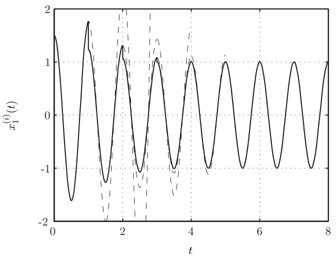


a) starting point I, the iteration converges to an unstable focus

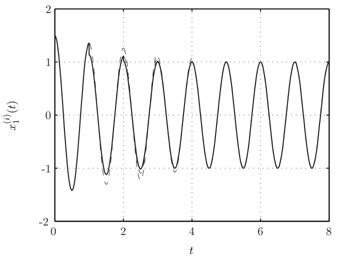
a) starting point II, the iteration converges to a stable focus

Figure 3.5: The Newton-Picard stabilization for the steady state continuation; solid line is a stabilized iteration process; dashed lines are unstabilized dynamic simulations

of the matrix F_x are summarized in Table 3.2.



a) starting point III, the iteration converges to an unstable periodic solution



b) starting point III, the iteration converges to a stable periodic solution

Figure 3.6: The Newton-Picard stabilization for the periodic orbit continuation; solid line is a stabilized iteration process; dashed lines are unstabilized dynamic simulations

Point	x_1^*	x_2^*	μ_1	μ_2
I	0.0	0.0	1.1052	1.1052
II	0.0	0.0	0.9048	0.9048
III	0.89	-0.45	1.2214	1.0
IV	0.89	-0.45	1.0	0.8187

Table 3.2: Results of the test examples I – IV

3.6 Conclusion

The numerical computation of periodic solutions of dynamical systems has been introduced in this chapter. The chapter at first concentrated on the definition of periodic orbits and a corresponding periodic boundary-value problem. The stability analysis of periodic orbits also has been presented in this chapter. The stability analysis is based on the Poincaré map definition and leads to the determining of the stability information of fixed points of the map. The non-hyperbolic fixed points of the Poincaré map are of great interest in applications, since such points specify stability changes. In Section 3.2 three possible non-hyperbolic fixed points with connection to the underlying periodic orbits have been presented.

The next topic that has been covered in this chapter is the parameter continuation of periodic orbits. The used method is the single shooting method that has been applied to the periodic boundary-value problem with additional phase and parametrization equations. The boundary-value problem with the Newton iteration method has been solved. This approach only can be used for quite small models, because the Newton method requires computation of the full $n \times n$ sensitivities matrix.

In the second part of the chapter the recursive projection method in the form of the fixed value problem for a map has been presented. The system under study is projected onto the low-dimensional generalized eigenspace of the linearization matrix, corresponding to the unstable and weakly stable modes, and its high-dimensional orthogonal complement. The high-dimensional subsystem is solved using an iteration method. In this work the Picard iteration is implemented, but in possible extensions of the RPM implementation more sophisticated methods can be used, such as BiCGSTAB or GMRES (BBC^{+}94). The low-dimensional subsystem is solved using a direct linear system solver. Information on the dominant eigenvalues in low-dimensional subspace also can be obtained from the splitting. The presented method will be efficient for a relatively small unstable subspace. Instead of computing n^2 sensitivities as with the full Newton method, only $n \times (n_p(1 + n_{sub} + n_{pic}))$ sensitivities are required. Additionally, $n \times (n_p n_{sub})$ and $n \times (n_p n_{pic})$ are required for two extra steps, namely, n_{sub} subspace iterations and n_{pic} iterations for the linear algebra solving with the Picard iteration.

After the recursive projection method introduction, three applications are presented. The first application is a linear algebraic system solution. The system is presented as a simple iteration method that only will converge for limited cases when the matrix $I - A$ has only eigenvalues with a magnitude less than one. The application of the stabilization procedure separates the solution vector into two subspaces that have eigenvectors as a basis. The first subspace is spanned by the eigenvectors of A that correspond to the eigenvalues with a magnitude less than one, such that the Picard iteration will converge in the subspace. The second one has a basis that is spanned by eigenvectors corresponding to other eigenvalues that cause slow convergence or divergence of the iteration method. The solution vector is updated with a direct method in this subspace. For such a low-dimensional subspace the method will require low computational costs. As an additional result the method computes dominant eigenvalues and corresponding eigenvalues for the matrix A.

The second and third application are continuation tasks for steady-state points and periodic solutions of dynamic systems. The system integration has been done as a “black-box” dynamic solver that allows to compute sensitivities and matrix-vector products Mv that are required for the iterative Newton-Picard method. Compared to an ordinary single shooting method, the Newton-Picard greatly reduces computational costs for high-dimensional simulation models with low-dimensional dynamic behavior.

It is important to note that the cost for one iteration step is proportional to the cost of one time integration. The amount of matrix-vector products with the monodromy matrix needed in the subspace and Picard iterations is independent of the number of degrees of freedom of the spatial discretization. Hence, if an efficient time integration code is available, the method particularly suits for problems that require a fine spatial discretization to capture the behavior of the physical system correctly and is very well suited to produce quantitative results, which usually require much finer discretizations than are required for the generation of qualitative results only. Additionally, the method computes dominant eigenvalues of F_x that can be interpreted in the stability analysis of fixed points.

Chapter 4

Case Studies

The present chapter contains examples of the nonlinear analysis of three chemical engineering models: a continuous flow stirred tank reactor model, a spatially distributed model of a high temperature fuel cell, and a crystallizer in continuous mode of operation with fines dissolution and classified product removal. These models show highly nonlinear behavior and can be used to present the nonlinear analysis algorithms implemented in this work.

The first example is a low dimensional model of a continuous flow stirred tank reactor. How the model sources can be generated and compiled in Diana simulation environment will be shown for this example. Also a basic dynamic simulation will be presented. The parameter continuation of a fixed point, the computation of the limit point variety with singular points detection, and the perturbation diagrams in a neighborhood of such points will be discussed in this example.

The second example is a spatially distributed molten carbonate fuel cell model. The destabilizing effect in this model leads to the formation of temperature hot spots. The nonlinear analysis helps to understand the formation of the hot spots to avoid damage of the fuel cell. The highly nonlinear behavior of this model is shown by the detected winged cusp points.

The last model represents a mixed-suspension-mixed-products removal crystallizer. The crystallizer is modeled by a discretized population balance equation and a mass balance equation. In recent studies (see, for example, (PK02)) periodic solutions in this model have been found. But due to the size of the model, traditional periodic solution continuation methods were not applicable in the past. It will be shown that the RPM algorithm in Diana is able to treat problem of such a high dimension easily.

4.1 Nonlinear analysis of a CSTR

For the continuous flow stirred tank reactor case study the well studied nonlinear benchmark problem of the iron(III)-catalyzed oxidation of ethanol with hydrogen peroxide to ethanal and acetic acid (ZMOG99) will be considered. Previous experimental and theoretical studies showed that this reactor can have a very complex steady-state and dynamic behavior (Haf68; ZMOG99; ZPMG00). The model shows highly nonlinear behavior and can display temperature and concentration oscillations, steady-state multiplicities, and Hopf bifurcation points.

The mass balances of the model read

$$\begin{aligned}
\frac{\mathrm{d}c_{\mathrm{H_2O_2}}}{\mathrm{d}t} &= \frac{\dot{q}_{\mathrm{in}}}{V}(c_{\mathrm{H_2O_2,in}} - c_{\mathrm{H_2O_2}}) - (r_1 + r_2 + r_3),\\
\frac{\mathrm{d}c_{\mathrm{CH_3CHO}}}{\mathrm{d}t} &= \frac{\dot{q}_{\mathrm{in}}}{V}(c_{\mathrm{CH_3CHO,in}} - c_{\mathrm{CH_3CHO}}) + (r_1 - r_2),\\
\frac{\mathrm{d}c_{\mathrm{CH_3COOH}}}{\mathrm{d}t} &= \frac{\dot{q}_{\mathrm{in}}}{V}(-c_{\mathrm{CH_3COOH}}) + r_2\\
\frac{\mathrm{d}c_{\mathrm{cat}}}{\mathrm{d}t} &= \frac{\dot{q}_{\mathrm{in}}}{V}(c_{\mathrm{cat,in}} - c_{\mathrm{cat}}) - (r_4 - r_5),
\end{aligned}$$

where c_i are concentrations, V is the liquid phase volume, $\dot{q}_{\mathrm{in}}$ denotes the volumetric feed flow, and $c_{i,\mathrm{in}}$ specify the feed concentrations.

The dependence of the reaction rates r_i on the reactor temperature T is described by the Arrhenius equation. The rate expressions reads

$$\begin{aligned}
r_1 &= k_1 e^{-E_1/(RT)} c_{\mathrm{cat}}\, c_{\mathrm{H_2O_2}}\\
r_2 &= k_2 e^{-E_2/(RT)} c_{\mathrm{cat}}\, c_{\mathrm{H_2O_2}}\, c_{\mathrm{CH_3CHO}},\\
r_3 &= k_3 e^{-E_3/(RT)} c_{\mathrm{cat}}\, c_{\mathrm{H_2O_2}},\\
r_4 &= k_4 e^{-E_4/(RT)} c_{\mathrm{cat}}\, \sqrt{c_{\mathrm{CH_3CHO}}},\\
r_5 &= k_5 e^{-E_5/(RT)} (c_{F,ges} - c_{\mathrm{cat}}).
\end{aligned}$$

where $c_{F,ges}$ is the total iron concentration. The pre-exponential factors k_i and energies of activation E_i can be found in (ZMOG99).

The energy balance reads

$$\begin{aligned}
V\rho c_p \frac{\mathrm{d}T}{\mathrm{d}t} &= \rho c_p \dot{q}_{\mathrm{in}}(T_{\mathrm{in}} - T) + (UA)_{\mathrm{cool}}(T_{\mathrm{cool}} - T) + V\sum_{i=1}^{3} r_i(-\Delta h_{\mathrm{R}})_i.\\
V_{\mathrm{cool}}\rho c_p \frac{\mathrm{d}T_{\mathrm{cool}}}{\mathrm{d}t} &= \rho c_p \dot{q}_{\mathrm{cool}}(T_{\mathrm{cool,in}} - T_{\mathrm{cool}}) + (UA)_{\mathrm{cool}}(T - T_{\mathrm{cool}}).
\end{aligned}$$

where $\dot{q}_{\text{cool}}$ is the volumetric coolant inlet flow, T_{in}, T_{cool} and $T_{\text{cool,in}}$ are the temperature of the feed flow, the temperature of the cooling water in the coil, and the temperature of the coolant inlet flow. $(UA)_{\text{cool}}$ signifies an overall heat transfer coefficient, which describes the energy transfer to the cooling coil. The values $(-\Delta h_{\text{R}})_i$ specify the reaction enthalpies under standard conditions, and ρc_p is a heat capacity of the liquid phase.

4.1.1 Dynamic simulation of a CSTR

The model source files from the MDL file (in this example the file name is `hafke.mdl`) can be produced with a command

```
mdl2diana Hafke_Reactor -f hafke.mdl
```

in a shell command line. The first argument is the model entity name, and the second one specifies the model file name after the command line flag `-f`. The compilation of the resulting C++ sources with the script `dianac`

```
dianac HafkeReactor
```

is performed.

Basic dynamic simulation of the CSTR can be demonstrated with the next script.

```
import sys, os
from diana import *
dm=GetDianaMain(sys.argv)
mm=dm.GetModelManager()
sf=dm.GetSolverFactory()

modelname='model/HafkeReactor.so'
model=mm.CreateModel(CAPE_CONTINUOUS, modelname)
model.Initialize()
eso=model.GetActiveESO()
epar=eso.GetRealParameters()
evar=eso.GetStateVariables()

solver=sf.CreateSolver(CAPE_DAE, model, 'ida.so')
solver.Initialize()
spar=solver.GetParameters()

ri=dm.CreateReportingInterface('basic')
solver.SetReportingInterface(ri)
ri.Add(spar['T'])
```

```
21 ri.Add(epar['qknormal'])
22 for var in evar: ri.Add(var)
23
24 par=({'qk': 4.0e-5, 'Tend':10000}, {'qk': 3.0e-5, 'Tend':20000},
25      {'qk': 1.0e-5, 'Tend':30000}, {'qk': 2.7e-5, 'Tend':60000})
26
27 spar['VerboseLevel'].SetValue(2)
28 spar['T0'].SetValue(0)
29
30 for p in par:
31     spar['Tend'].SetValue(p['Tend'])
32     epar['qknormal'].SetValue(p['qk'])
33     solver.Solve()
34
35 outdir='DynamicSimulation'
36 if not os.path.isdir(outdir): os.mkdir(outdir)
37 ri.WriteDataMatlab(outdir+'/Example.m')
```

Listing 4.1: The dynamic simulation of the CSTR in Diana

In line 1 of the script the Python and Diana modules are imported. The next three lines 2 – 4 initialize `DianaMain`, `ModelManager` and `SolverFactory` instances. The model manager `mm` loads the CSTR compiled model `HafkeReactor.so` in line 6, and in line 7 the model instance `model` is initialized. For convenience, in the lines 8 – 10 references to an ESO instance `eso`, real parameters `epar` and state variables collections `evar` are assigned. The IDA solver `solver` is loaded and initialized in the lines 12 – 14. After that, a reporting interface `ri` in the lines 16 – 20 for the independent variable t, the model parameter $\dot{q}_{\text{cool}}$, and the state variables is initialized. The lines 22 – 31 present a simple simulation scenario. In this scenario the volumetric coolant inlet flow $\dot{q}_{\text{cool}}$ changes the value during the simulation at the time points 10,000; 20,000; and 30,000. Finally, the results in the Matlab form in the lines 33 – 34 are saved to a result file.

Dynamic simulation results (Figure 4.1) show quite complicated behavior of the CSTR model. On the figure the model has either stable periodic or steady-state solutions for different values of the coolant inlet flow $\dot{q}_{\text{cool}}$ Such a behavior assumes existence of the Hopf point bifurcation for some values of the parameter $\dot{q}_{\text{cool}}$. This model will be used in further case studies to present capabilities of the nonlinear analysis tools in Diana.

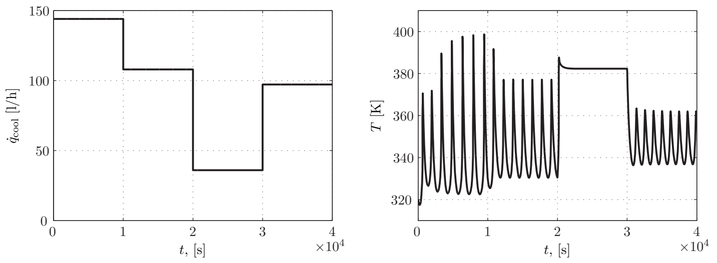


Figure 4.1: Simulation results of the CSTR for different values of the volumetric coolant inlet flow $\dot{q}_{\mathrm{cool}}$

4.1.2 Singularity Analysis of a CSTR

Previous experimental and theoretical studies showed that this reactor can show a very complex steady-state and dynamic behavior(Haf68; ZMOG99; ZPMG00). However, for this model higher order singularities have never been investigated before, mainly, because the derivation of the augmented equations system is very tedious even for such a comparatively small system. In the following, it will be shown that a singularity analysis can be done very easily using the DIANA nonlinear suite, and that such a singularity analysis is a highly efficient way to elucidate the nonlinear system behavior and to understand the process in a wide range of operation and model parameters.

The first step of the analysis consists in a continuation of steady-state solutions. The inlet temperature of the coolant $T_{\mathrm{cool,in}}$ is used as a continuation parameter in the following Python script (Listing 4.2). The script uses the preloaded ESO instance `model/eso` of the CSTR model and loads initial point data in line 1. In the lines 3 – 10 the solver instance `conti` for the steady-state parameter continuation (the shared library `sstate.so`) with the reporting interface `ri` are initialized. Some of the continuation parameters in the lines 12 – 16 are tuned. The inlet temperature of the coolant $T_{\mathrm{cool,in}}$ is marked as the continuation parameter in line 17. In addition, the method `AddFreeParameter` receives the lower and upper boundary values of the parameter. The actual continuation and storing of the results is done in the lines 18 – 19. During the continuation the solver computes linearized stability information along the solution curve and detects bifurcation points.

Under suitable conditions a hysteresis behavior as shown in Figure 4.2 exists.

```
eso.LoadState('InitialPoint.dat')

conti=sfactory.CreateSolver(CAPE_CONTI, model, 'sstate.so');
conti.Initialize();
cpar=conti.GetParameters()
ri=main.CreateReportingInterface('basic');
conti.SetReportingInterface(ri);
ri.Add(epar['tkzu']]);
ri.Add(evar['t']);
ri.Add(cpar['Stability']);

cpar['VerboseLevel'].SetValue(0)
cpar['MaxStepsAmount'].SetValue(5000)
cpar['MaxStepSize'].SetValue(5.0)
cpar['StabilityCheck'].SetValue(True)
cpar['ConditionCheck'].SetValue(SingularityNone)
conti.AddFreeParameter('tkzu', 250.0, 400.0)
conti.Continuate()
ri.WriteDataMatlab('OneParameterContinuation.m')
```

Listing 4.2: The steady-state continuation of the CSTR model

Such a hysteresis can be found easily in most cases, using either physical considerations or well-known graphical constructions (vH58). The solution branches in Figure 4.2 lose stability in Hopf bifurcation points (ZMOG99). This points can be a subject of a further analysis, for example a Hopf point continuation curve in Figure 4.3, however, this is not considered further in this work.

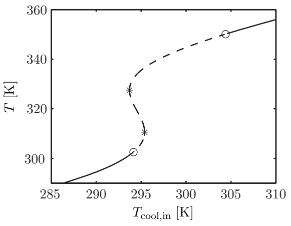


Figure 4.2: Initial one-parameter continuation of steady states; solid lines stand for stable solutions; dashed lines stand for unstable solutions; asterisks mark limit points and circles are Hopf bifurcation points

The two limit points in Figure 4.2, which border the interval of multiple steady states, change their position, if another model parameter, *e.g.,* the coolant flow

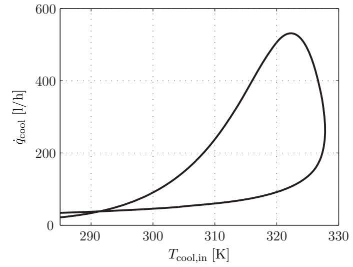


Figure 4.3: The two-parameter continuation of the Hopf point; the starting point is taken from the continuation in Figure 4.2 for the initial value of the volumetric feed flow $\dot{q}_{\text{in}} = 50.4$ [l/h]

rate $\dot{q}_{cool}$ is varied. The position of the two limit points as a function of the model parameters $T_{\text{cool,in}}$ and $\dot{q}_{cool}$ is depicted in the left diagram of Figure 4.4. The curve in the left diagram was generated by a two parameter continuation, using one of the limit points from Figure 4.2 as a starting value. In order to detect a singularity of the next higher codimension, the test function g_{zz} of Eq. 2.24 is evaluated along the solution branch (see right diagram in Figure 4.4). Eventually, this test function vanishes and indicates the occurrence of a codimension-1 hysteresis point (diamond box in Figure 4.4).

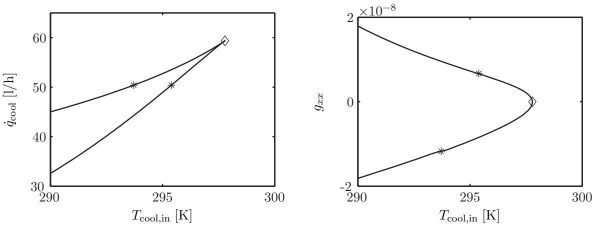


Figure 4.4: Left diagram: the limit point continuation, asterisks mark the limit points shown in Figure 4.2; right diagram: g_{zz} test function that indicates a hysteresis point when vanishing

This diagrams are plotted with the data produced by the following script (Listing 4.3). This script has minor changes with the previous from the previous one. The solver instance `sing` is loaded from the shared library `sanalyser.so`. Instead

```
eso.LoadState('LimitPoint.dat')

sing=sfactory.CreateSolver(CAPE_CONTI, model, 'sanalyser.so')
sing.Initialize()
spar=sing.GetParameters()
spar['ConditionShow'].SetValue(SingularityGxx)
ri=main.CreateReportingInterface('basic')
sing.SetReportingInterface(ri)
ri.Add(epar['tkzu'])
ri.Add(evar['temp'])
ri.Add(spar['Gxx'])

spar['VerboseLevel'].SetValue(2);
spar['MaxStepSize'].SetValue(5.0);
spar['InitialDirection'].SetValue(-1);
spar['ConditionEquations'].SetValue(SingularityGx);
conti.AddFreeParameter('tkzu', 250.0, 400.0)
sing.AddFreeParameter('qknormal', 0.5e-10, 5e-3);
ret=sing.Continuate();
ri.WriteDataMatlab('LimitPointContinuation.m');
```

Listing 4.3: The limit point continuation of the CSTR model

of the stability information, the reporting interfaces gets the value of the test function g_{xx} during the continuation. Also, the solver has two "free" continuation parameters that are changed along the continuation curve. In addition to the inlet temperature of the coolant $T_{\mathrm{cool,in}}$, the coolant flow rate $\dot{q}_{cool}$ is added. The parameter `ConditionEquations` is changed to the value `SingularityGx` that adds an equation $g_z = 0$ to the augmented system 2.33.

The exact value of the hysteresis point can be computed with an extension of the previous script. The script changes the value of the parameter `ConditionCheck` to `SingularityGxx`, that adds estimation of the test function g_{zz} along the continuation curve. The result of the continuation is checked in line 21. If the return value shows that the test function has changed the sign (the value `ContiOkTestFunction`) than an additional parameter $c_{\mathrm{H_2O_2,in}}$ and the equation $g_{zz} = 0$ are added to the solver in the lines 22 – 24.

Three parameter continuation can compute this hysteresis point as a function of three model parameters, *e.g.,* $T_{\mathrm{cool,in}}$, $\dot{q}_{cool}$, and the inlet concentration of hydrogen peroxide $c_{\mathrm{H_2O_2,in}}$. The result is shown in the left-hand diagram of Figure 4.5. Similar to the previous step, a new test function g_λ is evaluated in addition to the computation of the solution branch. In this continuation the λ

```
spar['ConditionCheck'].SetValue(SingularityGxx)
conti.AddFreeParameter('tkzu', 250.0, 400.0)
sing.AddFreeParameter('qknormal', 0.5e-10, 5e-3)
ret=sing.Continuate()
if ret==ContiOkTestFunction:
    sing.AddFreeParameter('c_zu[1]', 0.0, 10000.0);
    spar['ConditionEquations'].SetValue(SingularityGx|
                                        SingularityGxx)
    if sing.Solve()!=SolveSuccess:
        print 'Error in the hysteresis point calculation'
    else:
        eso.SaveState('Hysteresis.dat')
else:
    print 'Error in the hysteresis point estimation'
```

Listing 4.4: The hysteresis point location

is the coolant flow rate $\dot{q}_{cool}$ and is specified by the value of the solver parameter `LambdaParameter`. This allows to find the co-dimension 2 pitchfork bifurcation point marked by a square box in Figure 4.5, the organizing center for the chosen set of three free parameters. Before considering the relevance of the organizing center for the model behavior, one should note that this singularity is detected with a minimum input from the user. Each time a singularity is detected, the corresponding test function $g_z = 0$, $g_{zz} = 0$, $g_\lambda = 0$ is added automatically to the augmented equation system by setting the parameter `ConditionEquations` to a value `SingularityGx|SingularityGxx|SingularityGp`. All the user has to do is to choose another free model parameter for the next continuation run and to restart the continuation algorithm.

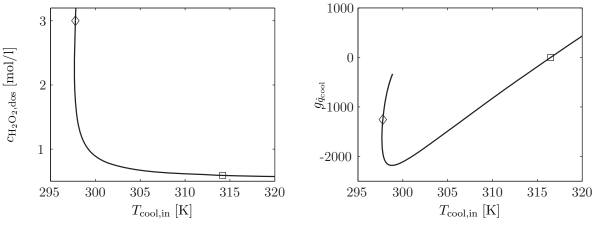


Figure 4.5: Hysteresises continuation and corresponding g_λ test function; square box is a pitchfork bifurcation point

The singularity theory predicts that in the vicinity of the organizing center all possible kinds of qualitative behavior of the system occur. This is illustrated by the bifurcation diagram in Figure 4.6. One can see that at the organizing center two curves of singularity varieties meet, which divide the parameter region in four domains I to IV of qualitatively different nonlinear behavior. Figure 4.7 shows typical one-parameter bifurcation diagrams for each of the four domains. In domain I, there are two solution branches: a low temperature branch that exists for all values of the coolant flow, and a high temperature branch that vanishes for coolant flows above $\dot{q}_{cool} \approx 0.75$l/h. When starting the reactor at the high temperature branch and slowly increasing the coolant flow, one will find a sudden temperature drop at $\dot{q}_{cool} \approx 0.75$l/h, as the system moves to a stable steady state at the lower branch. A further slow increase of the coolant flow first causes a gradual increase of the reactor temperature and later, at $\dot{q}_{cool} \approx 1.4$l/h a sudden jump of the reactor temperature, when one of the limit points on the lower branch is passed. In domain II, there is still one solution branch that vanishes for larger values of the coolant flow, and one branch that exists for all values of $\dot{q}_{cool}$. In contrast to domain I, this second branch is now completely stable. Domain I and II are separated by a transcritical bifurcation variety. This means that at the border between the two domains, the two solution branches meet and form a fork-like structure. The border between domain II and domain III is a hysteresis variety, *i.e.,* domain III possesses an additional hysteresis compared to domain II. This can be seen from Figure 4.7 c). Finally, a transition from domain III to domain IV causes the unstable part of the lower solution branch from domain III to connect with the left-hand side part of the upper solution part. The stable part of the lower solution branch and the right-hand side part of the upper solution part form another completely stable solution branch in domain IV.

From the discussion of Figure 4.6 and Figure 4.7 it should become clear that the organizing center contains system information in a very condensed manner. Once the organizing center is known, it is sufficient to determine steady-state solutions at a few points in the parameter space in order to understand the parameter dependent steady-state behavior of the system completely. From Figure 4.6 and 4.7, one can read the qualitative behavior of the CSTR for any value combination of T_{cool}, $\dot{q}_{cool}$ and $c_{\mathrm{H_2O_2,in}}$ without having to compute the solutions explicitly.

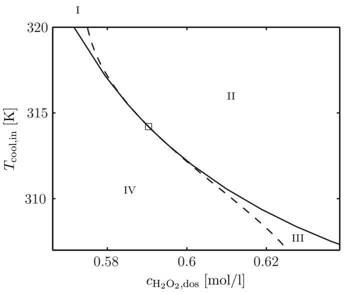


Figure 4.6: Singularity varieties of the CSTR model; the solid line is hysteresis variety, the dashed line is transcritical bifurcation variety, the square box is the pitchfork bifurcation point

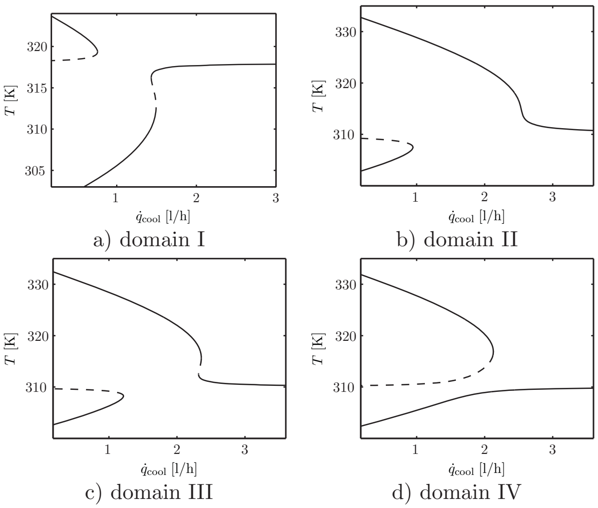


Figure 4.7: The one-parameter continuation of steady states with $\dot{q}_{\text{in}}$ as continuation parameter; solid lines are stable solutions, dashed lines are unstable solutions

4.2 Singularity analysis of a Molten Carbonate Fuel Cell

As the second example of the singularity analysis tool, a spatially distributed model of a high temperature fuel cell is considered. The control of the cell temperature is one of the most important issues for the process operation of high temperature fuel cells. Temperature maxima, temperature minima and spatial temperature gradients have to be kept within strict limits in order to avoid damage of the fuel cell. Therefore, a thorough understanding of the formation of hot spots is crucial for the development of process control strategies. Such understanding can be gained from singularity analysis, as will be shown in the following.

An analysis of this model shows that nonlinear effects are mainly caused by the temperature dependent electrical conductivity of the electrolyte in combination with the exothermic electrochemical reaction. The effect can occur in solid oxide fuel cells (SOFCs) as well as in MCFCs. It is illustrated in Figure 4.8. The charge

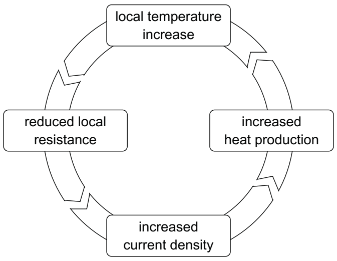

local temperature increase
increased heat production
increased current density
reduced local resistance

Figure 4.8: Temperature dependent electrical conductivity causes thermal instabilities

transport in the electrolyte of a high temperature fuel cell is mainly accomplished by migration of ions. Therefore, the electrical conductivity of the electrolyte increases with increasing temperature (*e.g.,* (YRSM+04)). This property may be a potential source of instability: A local temperature rise reduces the resistance of the electrolyte and hence increases the local current density. As the current density is directly coupled to the reaction rate, the reaction rate and hence the reactive heat production increase locally. This causes a further temperature rise, i.e. the temperature disturbance is amplified. It will be shown later that this mechanism may narrow down the part of the electrolyte that actually transports charge. Channels of high current density may form in the electrolyte, and hot spots will result. The generation of electrical current may be reduced to narrow channels with very high temperatures (MKS04; MKS06).

4.2.1 MCFC model

A spatially one-dimensional model is considered. The space coordinate is in the direction of the main coordinate of the electrode area, as shown in Figure 4.9. The main model assumptions are:

- Heat conductivity of the electrodes and the electrolyte in η direction is described by Fourier's law.
- The electrochemical reactions on anode and on cathode side are of the Butler-Volmer type.
- It is assumed that the gas composition in the anode bulk and in the cathode bulk is constant, *i.e.,* low fuel utilization and negligible concentration polarization is assumed.
- In the electrolyte, charge is transported only perpendicular to the η coordinate. The electrical conductivity of the electrolyte is temperature dependent and is described by an Arrhenius function.

A detailed derivation of the model was given in (MKS04).

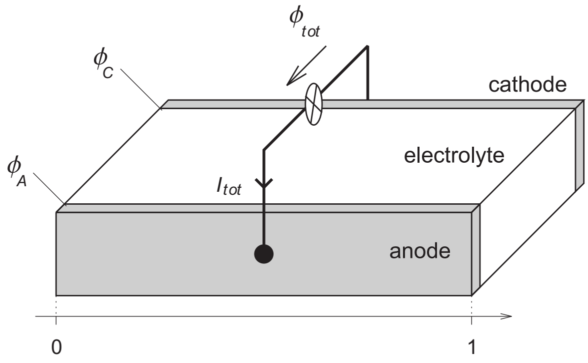


Figure 4.9: Scheme of the high temperature fuel cell considered in this work

The resulting dimensionless model equations read

$$\begin{aligned}
\frac{\partial\Theta}{\partial\tau} &= \frac{\partial^2\Theta}{\partial\eta^2} + \left(B - \phi^{tot}\right) i' - Bi_1\Theta && (4.1)\\
\left.\frac{\partial\Theta}{\partial\eta}\right|_{0,\tau} &= Bi_2\Theta(0,\tau) && (4.2)\\
\left.\frac{\partial\Theta}{\partial\eta}\right|_{1,\tau} &= -Bi_2\Theta(1,\tau) && (4.3)\\
i' &= \psi^A \exp\left(\gamma^A \frac{\Theta}{1+\Theta}\right)\left\{\exp\left(-(1-\beta^A)\gamma^{eq}\frac{\phi^A}{1+\Theta}\right) - \right.\\
&\qquad \left. -K_{eq}^A \exp\left(\beta^A\gamma^{eq}\frac{\phi^A}{1+\Theta}\right)\right\} && (4.4)\\
i' &= \psi^C \exp\left(\gamma^C \frac{\Theta}{1+\Theta}\right)\left\{\exp\left(-(1-\beta^C)\gamma^{eq}\frac{\phi^C}{1+\Theta}\right) - \right.\\
&\qquad \left. -K_{eq}^C \exp\left(\beta^C\gamma^{eq}\frac{\phi^C}{1+\Theta}\right)\right\} && (4.5)\\
i' &= \psi^E \exp\left(\gamma^E \frac{\Theta}{1+\Theta}\right)\left(\phi^A + \phi^C - \phi^{tot}\right) && (4.6)\\
I &= \int_0^1 i' d\eta && (4.7)
\end{aligned}$$

The unknowns in this model are the cell temperature Θ, the current density i, the potential difference on anode ϕ^A and on cathode ϕ^C side and the total cell voltage ϕ^{tot}. The first three lines of the equation system above contain the energy balance and the corresponding boundary conditions. Equations 4.4 and 4.5 give a correlation for i due to the Butler-Volmer reaction kinetics. Equation 4.6 states Ohm's law for the electrolyte. The last line is an overall charge balance of the cell that sums up the local current densities to a total cell current.

4.2.2 Cell operation at constant voltage

This section considers the steady state behaviour of the fuel cell, if the cell voltage is kept fixed. This mode of operation simplifies the mathematical analysis, as the integral equation (4.7) can be considered as an explicit equation for I and does not have to be solved for one of the other variables.

System of infinite length

In a first step, the steady state solutions for a system of infinite length are studied. The steady state version of (4.1) can be transformed to the following system of two ordinary differential equations:

$$\frac{d\Theta}{d\eta} = \Theta_p \tag{4.8}$$

$$\frac{d\Theta_p}{d\eta} = (\phi^{tot} - B)i' + Bi_1\Theta \tag{4.9}$$

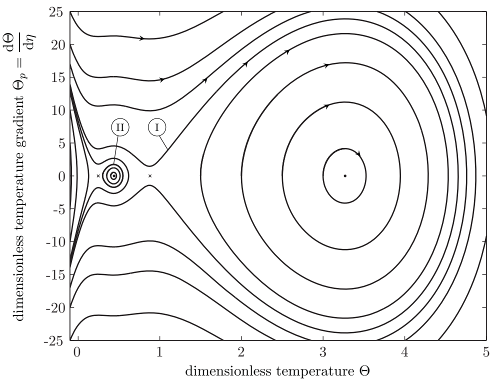
The newly introduced variable Θ_p is the derivative of the temperature profile with respect to space or, in other words, the spatial temperature gradient. For a given total cell voltage ϕ^{tot}, the steady state system is completed by the three implicit algebraic correlations (4.4-4.6) for the unknowns i', ϕ^A, ϕ^C. A phase plane analysis can be applied to the steady state system: For different solutions, the temperature gradient is plotted against the temperature at the same point. The independent variable η is eliminated and appears as an arclength parameter in this representation of the solution. A typical result is shown in Figure 4.10. Solutions with a spatially constant temperature profile are points on the $\Theta_p = 0$ axis or equilibrium points of (4.8,4.9). Those equilibrium points are either saddle points or marginally

B	$=$	0.9328
$\psi^A = \psi^C$	$=$	$7.385 \cdot 10^4$
$\gamma^A = \gamma^C$	$=$	15.04
γ^{eq}	$=$	32
ψ^E	$=$	$4.86 \cdot 10^3$
γ^E	$=$	1.04
Bi_1	$=$	6480
Bi_2	$=$	1.08
β^A	$=$	0.5
K_{eq}^A	$=$	10^{-7}

Table 4.1: Parameter values used in the simulations of MCFC

stable focuses, as follows directly from the Jacobian

$$J := \begin{pmatrix} 0 & 1 \\ Bi_1 + (\phi^{tot} - B)\frac{\partial i'}{\partial \Theta} & 0 \end{pmatrix} \tag{4.10}$$



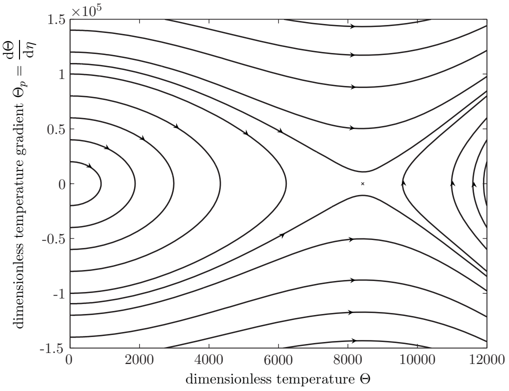


Figure 4.10: Steady solutions of an infinite length system for constant cell voltage; $\phi^{tot} = 0.425$, other parameters as in Table 4.1; $\times$-labels are saddle points and $\cdot$-labels are marginally stable focuses (centers)

The spatially inhomogeneous solutions can be grouped into two categories. The first category are solutions coming from and going towards infinity, *e.g.,* solution "I" in Figure 4.10. They are characterised by a single temperature maximum, the utter right point of curve "I". The second category are solutions on closed trajectories, *e.g.,* solution "II" in Figure 4.10. Those solutions are periodic in space and lead to the formation of temperature patterns and current density patterns: When moving along the space coordinate, the temperature oscillates between a minimum temperature (the utter left point of curve "II") and a maximum temperature (the utter right point of curve "II"). As the closed trajectories surround focuses, the existence of a focus point is a prerequisite for pattern solutions. Figure 4.11 shows the location of saddle node bifurcations in the γ^E-ϕ^{tot} parameter plane. The saddle node bifurcations denote parameter values, where a focus and a saddle point coincide. Focuses and pattern solutions exist in the shaded region of Figure 4.11. Obviously, the phenomenon of pattern solutions can be found in a large area of

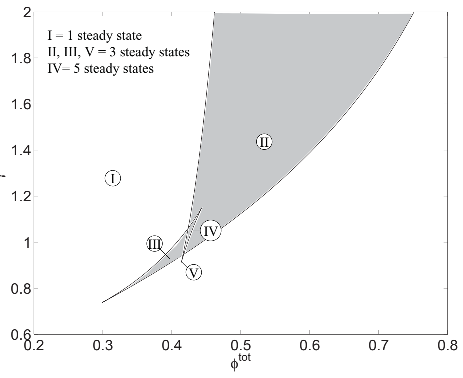


Figure 4.11: Saddle node bifurcations of an infinite length system in the γ^E-ϕ^{tot} plane, parameters as in Table 4.1

parameter values.

System of finite length

The phase plane analysis is extended now to a system of finite length that has to fulfil the boundary conditions (4.2, 4.3). In a phase diagram, the left and the right boundary condition are straight lines with slope Bi_2 and $-Bi_2$, respectively. They are shown as dashed lines in Figure 4.12. Steady state solutions of the finite length

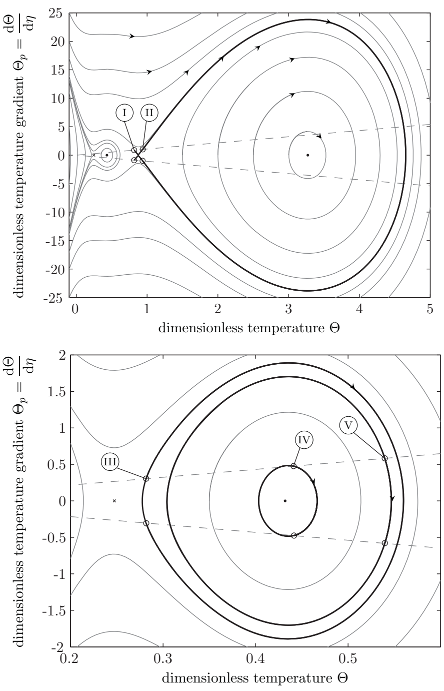


Figure 4.12: Steady solutions of an finite length system for constant cell voltage; $\phi^{tot} = 0.425$, other parameters as in Table 4.1; dashed lines are points fulfilling the boundary conditions; circles on the trajectories mark boundary points of the solutions

system always start on the upper dashed line, follow one of the solution trajectories of the infinite length system, and end on the lower dashed line. The distance in space covered while moving from the starting point to the end point must be equal to the length of the system. If the system is long enough, a solution may take several turns on a closed trajectory before reaching the right boundary point. Due to those additional conditions, only a subset of the solutions of the infinite length system are solutions of the finite length system, as well. In Figure 4.12, the solutions of the finite length system are indicated by bold lines. The corresponding temperature profiles are shown in Figure 4.13. In this example, five solutions are

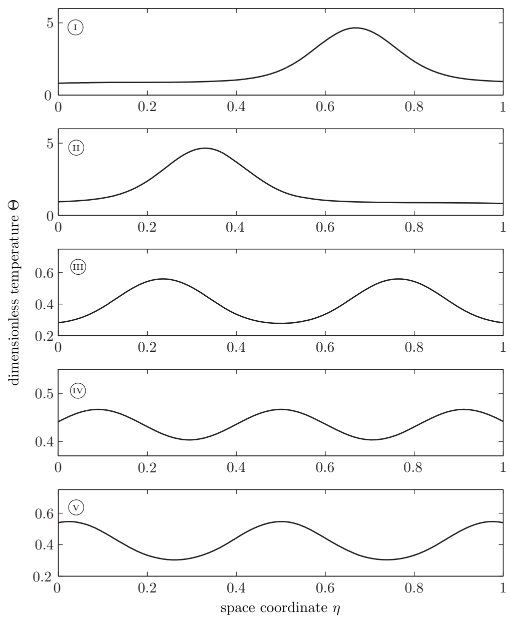


Figure 4.13: Spatial temperature profiles of the solutions I – V in Figure 4.12

found to coexist. The solutions III – V are periodic pattern solutions moving on closed trajectories of the infinite length system. They are symmetric with respect to the centre of the considered space interval. But also asymmetric solutions are possible, as can be seen from solution I and II. Both solutions run along the same trajectory of the infinite length system, but start and end at different boundary points.

The described method to construct solutions of the finite length system is quite convenient, if the cell voltage is a given parameter. However, it is hardly applicable to the case of constant cell current. Further, one does not obtain information on the stability of the found solutions. Therefore, in the next section numerical bifurcation analysis will be applied to the system.

4.2.3 Cell operation at constant current

Usually, fuel cells are characterised by current-voltage plots, which show the voltage response of a cell to a given current. Therefore, in this section the cell behaviour is analysed numerically for the case of constant cell current. A cell with a finite length is considered. After spatial discretization, the model consists of 201 algebraic and 200 ordinary differential equations. The scripts for the calculation of continuation curves are similar to presented in the previous section and are omitted here.

The purpose of the nonlinear analysis is to show that in high temperature fuel cells thermokinetic instabilities may result from the temperature dependent electrical conductivity of the electrolyte. An example for this behavior is shown in Figures 4.14 – 4.17. Figure 4.14 shows a pitchfork variety in the $\gamma^A - Bi_2$ parameter plane. The winged cusp point, which is characterized by vanishing test functions g, g_z, g_{zz}, $g_{I_{tot}}$, $g_{zI_{tot}}$, was located in a similar way as described in the previous example. The point is shown as a triangle mark in Figure 4.14.

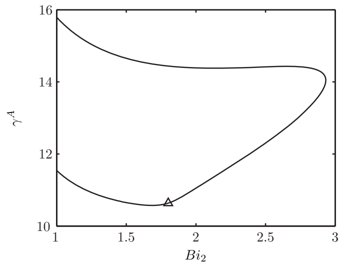
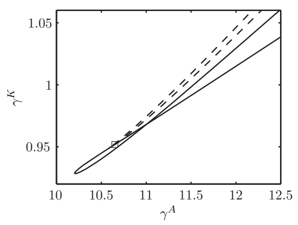
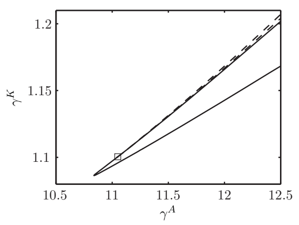


Figure 4.14: The pitchfork variety of the fuel cell model; triangle is the winged cusp point

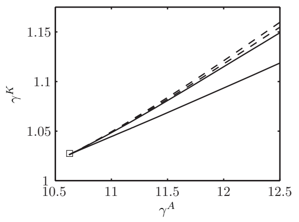
Figure 4.15 shows codimension-1 and codimension-2 singularities in the vicinity of the found winged cusp point. Figure 4.16 shows a bifurcation diagram with the cell current as a bifurcation parameter. Parameters γ^K, γ^A and Bi_2 are chosen in a parameter domain close to the winged cusp. For the magnified region of interest on the right-hand side of Figure 4.16 steady-state profiles are presented. The bifurcation diagram is typical for the complexity of the system behavior. For small values of the cell current $I_{tot} < 8678.5$ (Figure 4.17a), the steady state solution is always stable and unique. At the critical cell current value $I_{tot} \approx 8678.5$ two limit



a) $Bi_2 = 1.5$



b) $Bi_2 = 2.0$



c) $Bi_2 = 1.796$ (singular case)

Figure 4.15: Singularity varieties of the fuel cell model; solid lines are hysteresis varieties, dashed lines are isola or transcritical bifurcation varieties, square boxes are pitchfork bifurcation points

points appear simultaneously. Therefore, five steady states coexist in the interval $8678.5 < I_{tot} < 12991$ between the limit points and a subcritical pitchfork point (Figure 4.17b). In the next interval $12991 < I_{tot} < 15500$ that is limited by two subcritical pitchfork points (Figure 4.17c) the internal temperature peak becomes unstable and the system has two stable steady states with high temperatures on the left or right borders.

Interestingly, the nonlinear phenomena presented so far have no direct relation to the winged cusp point. This point is responsible for the appearance of a hysteresis in the interval $36524 < I_{tot} < 37255$. The hysteresis causes the appearance two additional unstable steady states and the model acquires five co-existing steady states for $I_{tot} = 3.7 \cdot 10^4$ (see Figure 4.17d). The two stable solutions possess a high temperature peak at the right or at the left system boundary and hence are no feasible operation points. The three unstable solutions have a moderate temperature maximum in the middle. This is a very useful result for the process operation. If the unstable steady states are stabilized by a suitable feedback control, it is possible to operate the fuel cell at much higher currents with a lower temperature level inside the cell. Obviously, this insight would hardly have been obtained without the tool of numerical singularity analysis.

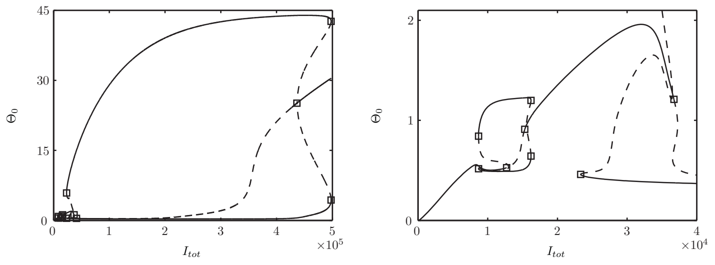
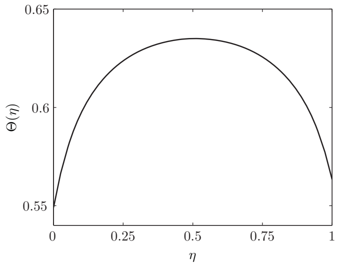
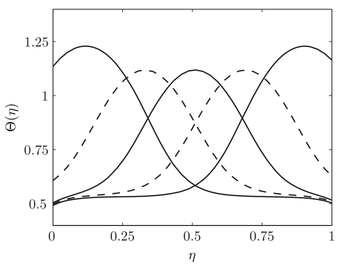


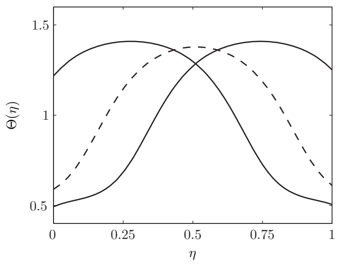
Figure 4.16: The steady-state temperature dependence $\Theta(I_{tot})$ and a magnified region of interest



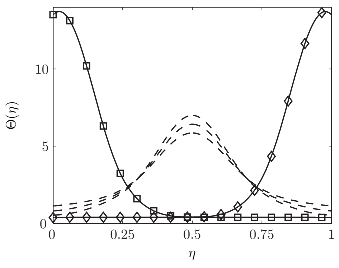
a) $I_{tot} = 8000$



b) $I_{tot} = 10000$



a) $I_{tot} = 14000$



b) $I_{tot} = 37000$

Figure 4.17: Steady-state profiles $\Theta(\eta)$ for parameters $\gamma^K = 1.02$, $\gamma^A = 12.0$, $Bi_2 = 1.5$ (solid lines with different marks correspond to different stable steady states)

4.3 Periodic solutions in a MSMPR Crystallizer

Periodic oscillations in crystallization processes have been found by various authors in theoretical and experimental studies ((RL88), (PK02)). Such results are important in applications, because the oscillations should be avoided for a better product quality. Conversely in special cases the periodic oscillations may be useful for industrial applications (RMK+06; SRP+98). A continuous mixed suspension mixed product removal (MSMPR) crystallization process is considered as an example. The process is described by a population balance model including a fines dissolution and a classified product removal, which are possible sources of nonlinear behavior (PK02). A crystallizer in continuous mode of operation with the fines dissolution and a classified product removal is shown in Figure 4.18. The crystallization process is modeled under the following assumptions:

- ideal mixing
- isothermal operation
- constant overall volume (liquid+solid)
- nucleation of crystals of negligible size
- size-independent growth rate
- no particle breakage, attrition or agglomeration.

The population balance equation for the particle size distribution $F(L,t)$ is

$$\frac{\partial F}{\partial t} = -\frac{\partial (GF)}{\partial L} - \frac{q}{V}(h_f(L) + h_p(L))F \tag{4.11}$$

with the boundary condition

$$F(0,t) = \frac{B(c,t)}{G(c,t)} = \frac{k_b(c(t) - c_{\text{sat}})^b}{k_g(c(t) - c_{\text{sat}})^g}. \tag{4.12}$$

The classification functions specifying the fines dissolution and the product classification are given by

$$\begin{aligned} h_f(L) &= R_1(1 - h(L - L_f)), \\ h_p(L) &= 1 + R_2 h(L - L_p), \end{aligned} \tag{4.13}$$

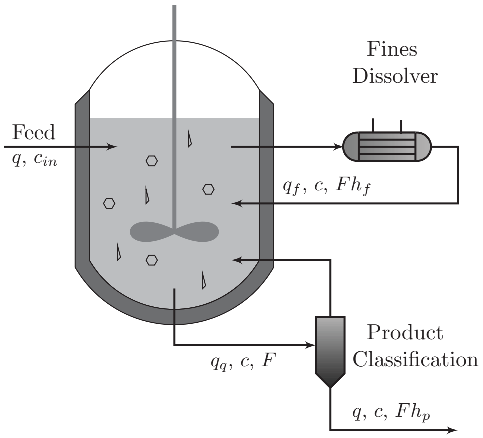


Figure 4.18: Continuous crystallizer with fines dissolution and classified product removal

where $h(L)$ is a unit step function and $R_1 = q_f/q$ and $R_2 = (q_q - q)/q$ are the recycle ratios.

The mass balance of solute is

$$M_A \frac{\mathrm{d}c}{\mathrm{d}t} = \frac{q(\rho - M_A c)}{V} + \frac{\rho - M_A c}{\varepsilon} \frac{\mathrm{d}\varepsilon}{\mathrm{d}t} + \frac{q M_A c_{in}}{V\varepsilon} - \frac{q\rho}{V\varepsilon} \left(1 + k_v \int_0^\infty (h_p(L) - 1) F L^3 \, \mathrm{d}L \right) \tag{4.14}$$

where ε is the void fraction which is given by

$$\varepsilon = 1 - k_v \int_0^\infty F L^3 \, \mathrm{d}L.$$

For the numerical simulation the population balance model (4.11) and (4.14) has to be converted to a set of differential algebraic equations. For the population balance of the dispersed phase a simple finite difference approximation on a non-equidistant but fixed grid is used according to

$$z_i = L_{max} \left(\frac{i}{N} \right)^3, \quad i = 0, 1, \ldots, N$$

with maximal length of crystals L_{max} and a total number of grid points N.

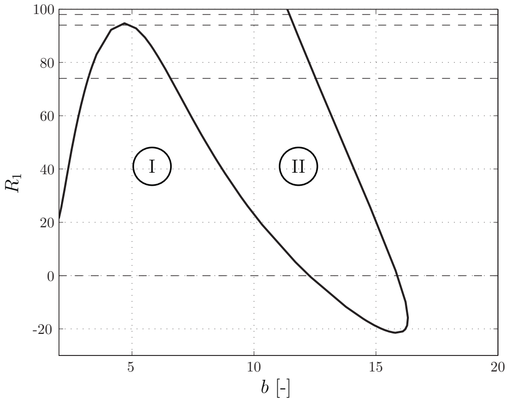
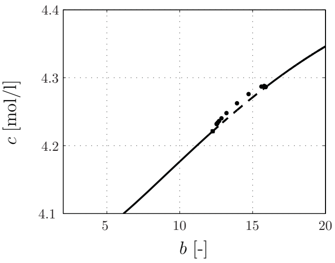


Figure 4.19: Hopf points continuation and instability region in the R_1/b plane (results for unphysical negative values of R_1 are added for the sake of completeness)

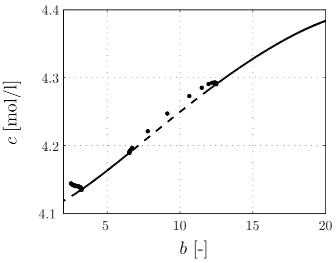
The model parameters can be found in (PK02). The population balance equation (4.11) of the model is discretized with 200 grid points.

To locate a region with periodic solutions the Hopf point continuation is performed. The results are shown in Figure 4.19. The region II presents a parameter domain in the R_1/b plane with oscillatory behavior. As shown in the figure for a fixed reflux ratio R_1 it is possible to have domains without oscillations, and domains with one or two oscillatory regions. In Figure 4.20 periodic continuation diagrams computed with the RPM are presented. Four sub-figures in Figure 4.20 correspond to the dashed lines on the Hopf points diagram for different reflux ratios R_1. In Figure 4.21 corresponding diagrams with the "slowly convergent" Floquet multipliers are presented. For $\varrho = 0.9$ there are only two of these multipliers in the sense of ordering (3.30). Consequently, the restricted monodromy matrix $V_p^T M V_p$ has a low dimension (2×2). Treating this small subspace by Newton iteration is of course much faster then applying Newton iteration to the whole n-dimensional model, for example, computation of a periodic solution branch for the MSMPR example with 200 grid points requires in average two hours with the RPM reduction and two and half days without it.

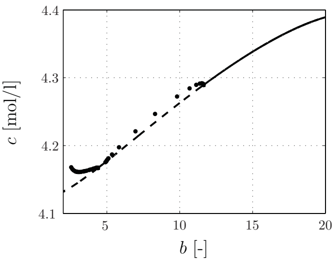
In Figure 4.22 results of the dynamic simulation of the model for some parameters are presented. Figure 4.22c shows result for the periodic solution branch



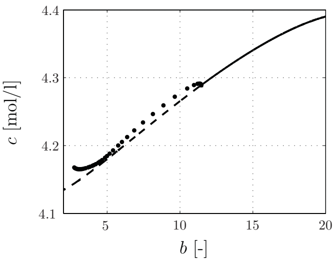
a) $R_1 = 0$



b) $R_1 = 75$

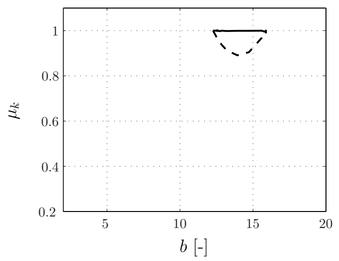


a) $R_1 = 94$

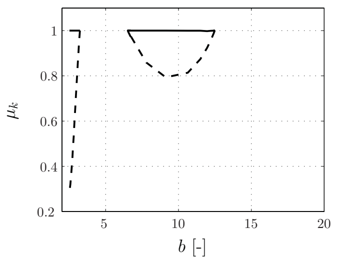


b) $R_1 = 98$

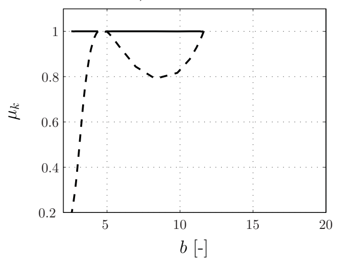
Figure 4.20: The periodic orbits continuation for different values R_1 in Figure 4.19



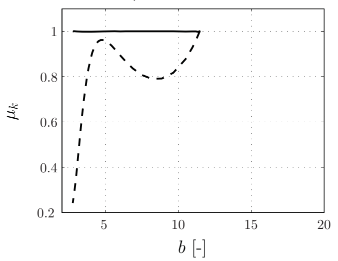
a) $R_1 = 0$



b) $R_1 = 75$



a) $R_1 = 94$

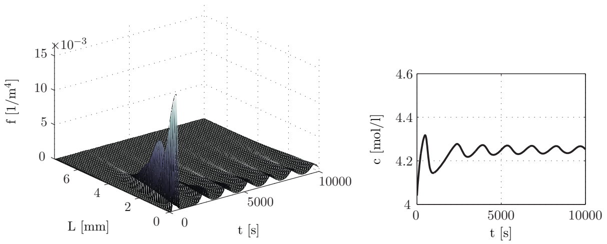


b) $R_1 = 98$

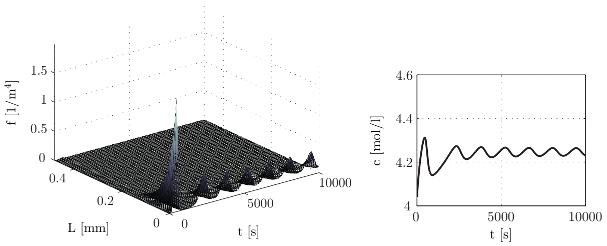
Figure 4.21: Corresponding Floquet multipliers to the diagrams in Figure 4.20 (the solid line shows the trivial unit multiplier and the dashed line is the dominant one)

for the smaller value of the nucleation exponent b in Figure 4.20b. The solution branch terminates at the region for lower values of the exponent due to the numerical failure of the dynamic solver `DASPK`. For such values the left boundary of the population distribution function f reaches high values and the solver returns a computational error.

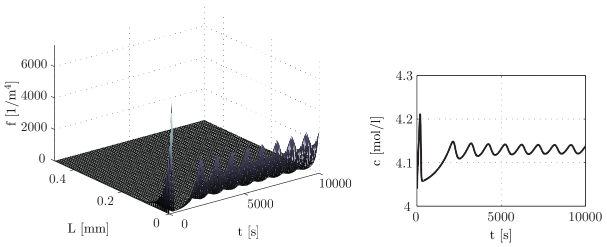
This example presents the virtues of the recursive projection method applied to the continuation of the periodic solutions for large systems. Also the numerical calculation shows that the continuous mixed suspension mixed product removal crystallization process with growth and nucleation terms has low-dimensional dynamics. This property of the reactor model can be used for the development of efficient numerical codes for the optimization, parameter estimation, or control of the model.



a) $b = 14,\ R_1 = 0$



b) $b = 10,\ R_1 = 75$



c) $b = 3,\ R_1 = 75$

Figure 4.22: Results of the dynamic simulation for different parameter values

Chapter 5

Conclusion and Future Directions

5.1 Conclusion

The majority of chemical engineering processes is characterised by strong nonlinearities. Many studies, including the three examples of this work, show that singularity analysis can help to understand the parameter dependent behavior of chemical processes much better, to detect undesired system states caused by nonlinearities, and to make use of nonlinear effects for improved process operation or process design. As chemical processes typically contain a large number of free parameters, the nonlinear behavior in a high-dimensional parameter space has to be characterized, and a tool is desirable that does this characterization efficiently and without much input from the user side.

In this thesis, a family of nonlinear analysis tools in the software package Diana has been presented. The software developed in this project is able to generate automatically augmented equation systems for the analysis of higher order singularities and can be applied to complex models of high system order. The user only is asked to provide his model equations in a symbolic form. The software consists of two separate tools: the package ProMoT for the symbolic manipulation of the model and for the generation of a C++ code containing the model equations, and the numerical solution engine Diana. The model interface of Diana follows the CAPE-OPEN standard. Therefore, the continuation and solution algorithms in Diana can be applied to any model with a CAPE-ESO interface. This makes the tool sufficiently open to other simulators.

This thesis makes several contributions in numerical nonlinear analysis and applications in bifurcation analysis:

- The basic object-oriented software architecture for the dynamic simulation and nonlinear analysis tool Diana has been implemented. The architecture includes a definition of simulation models that presents continuous dynamic systems in the implicit differential-algebraic equations form. Also the symbolic differentiation with a variable order of the model functions with respect to the state variables and the model parameters has been provided. The model interface of the choice is the CAPE-OPEN standard interface DAESO. This interface allows to use the defined model with third-party simulation environments via object request mechanisms, like, CORBA or DCOM. The software architecture also contains solver interfaces that allow to make numerical simulations or analysis of models. In this interface suite the differential-algebraic equation solver based on a BDF solver IDA/Sundials has been implemented.

- In addition to the CAPE-OPEN standard, the extended interfaces for the parameter continuation and bifurcation analysis have been introduced. The realization of this interfaces includes a predictor-corrector continuation solver that is used as a base abstract class. The solver computes a one-dimensional solution curve of a nonlinear subproblem that is defined in a derived solver. In this work three subclasses of the parameter continuation solver have been implemented:
  - The first solver locates steady-state points of a dynamical systems for a given initial state and performs the continuation of such points with respect to the specified parameter. Along the steady state continuation curve the solver can compute generalized eigenvalues of the model linearization. The eigenvalues deliver stability information of the computed steady state. Also the critical points, when the real parts of the eigenvalues change a sign, can be detected.
  - Another solver allows to compute varieties of limit steady-state points. The solver reduces the model to a scalar equation in a one-dimensional subspace that belongs to a zero eigenvalue of the steady-state linearization. This is performed with the help of the Lyapunov-Schmidt reduction. In addition, the solver computes derivatives of the reduced scalar equation that form the basis for the singularity analysis of the limit point variety. The set of implemented derivatives provide capabilities for the detection and continuation singular points, like, hysteresis, pitchfork, or winged cusp points.
  - The last solver forms the augmented system for the Hopf point com-

putation and the parameter continuation of such points. Monitoring of eigenvalues along a Hopf point curve gives possibility to detect degenerate Hopf points.

- Another issue that has been covered in this work is the parameter continuation of periodic solutions. Periodic solutions are computed using single shooting, which can be applied to high-dimensional systems with the recursive projection method. In addition to the original works in (SK93; Lus97) coupling terms that are important for non-normal problems have not been neglected. The implementation of the RPM is similar to the one described in (Lus97): the use of multiple varying Picard steps at each Newton-Picard step, the use of orthogonal subspace iteration with projection instead of ordinary orthogonal subspace iteration and a different and more robust strategy to determine the basis size.

- The three model examples in this work form an important part of it — they illustrate how implemented methods are used in applications.

That there are still many open questions has been pointed out at the end of the chapters of this work. The proposed possible extensions of the current work will be presented in the next section.

5.2 Suggestions for future work

In this section several aspects that deserve further attention for future work will be shown. It is clear that much work remains to be done in the area of the numerical nonlinear analysis. In addition to the implemented features, the following nonlinear problems can be subjects for future work:

- the implemented derivatives of the reduced equation (2.13) give possibility to find only hysteresis, pitchfork, or winged cusp organizing centers (see Figure 2.5). With additional derivatives g_{zzz}, g_{zzzz}, more degenerate points, like, asymmetric cusp or quartic fold, can be detected.

- in Section 2.10 the degenerate Hopf bifurcation points have been presented. Namely, generalized Hopf, Bogdanov-Takens, zero-Hopf, and double-Hopf points. The computation of such points is of great importance in various applications, due to a diverse variety of nonlinear phenomena in neighborhoods of such points. Such points indicate appearance of saddle-focus homoclinic and heteroclinic orbits, quasi-periodic and chaotic motions in a dynamical system.

- for engineering applications it is quite important to analyze parameter dependency and bifurcations of fixed points in discrete maps that present discrete dynamical systems. One can develop a discrete dynamical system extension for the Diana simulation environment. The application of the recursive projection method as the reduction procedure will give possibility to analyze high-dimensional discrete models that possess "low-dimensional" dynamics.
- enhancement of the recursive projection method can also be a topic for future work. For example, the Picard step in algorithm 2 can be substituted with other iterative methods, like GMRES or BiCGSTAB (BBC$^+$94). The power iteration step for the computing of eigenvalues can also be replaced by more sophisticated methods, for example, the implicitly restarted Arnoldi method (LSY98).
- the implementation of augmented systems for bifurcation analysis of codimension-0 bifurcations of periodic orbits may be also an interesting task. Such bifurcations determine stability domains of periodic solutions and appearance of new periodic branches. Another capability to increase stability and accuracy of the periodic solution and their bifurcations computation is the solving of periodic boundary-value problems (3.7) with more sophisticated methods, like, multiple shooting or collocation schemes (AMR95).
- recent results in numerical bifurcation analysis show great interest in computation and continuation of global phenomena, like, homoclinic, heteroclinic, point-to-periodic, or periodic-to-periodic orbits. The significant progress has been made in the continuation of homoclinic and heteroclinic connections involving cycles. A method based on projection boundary conditions for computing point-to-periodic and periodic-to-periodic connections has been introduced in (DR04). Another work (DKKvV07) also presents an algorithm for computation of point-to-periodic orbits. Such a global bifurcation analysis allows to understand many interesting phenomena in dynamical systems, like, the occurrence and disappearance of chaotic behavior.
- The continuation methods can also be extended to computing 2- or 3-dimensional manifolds with the multiple parameter continuation algorithm (Hen03).

- The usability of the software may be further increased by graphical user interface. So a better interface based on a graphical environment is highly required. The graphical user interface may consist of two parts. There should be a database to store computed solution paths, data on the convergence of the solver and bifurcation diagrams. Furthermore, a graphical front-end is needed that serves several purposes: to visualize the results, to prepare the input for the next run and to manage the database. The graphical user interface should assist in the choice of good defaults for the various parameters in the algorithm and be useful for both, "expert users" who understand the underlying algorithms and are capable to choose good values for all parameters of the algorithms, and "normal users" who have little understanding about the underlying code. The front-end should give possibility to visualize different types of plots, like bifurcation diagrams, phase plots, and specific system information, like, eigenvalues of a system linearization or characteristic multipliers, *etc.* Also the user feedback in the front-end is required. The user can stop a continuation and start another one from the desired initial point, or even start another continuation type based on the previous results. For example, the periodic orbits continuation can be started from the Hopf point, or the hysteresis point variety continuation can be started from the hysteresis point on a limit point continuation curve.

Appendix A

Software Architecture

The generic interfaces for solvers and models in DIANA are based on the specifications developed by the CAPE-OPEN community (JKB+99). These interfaces are designed to be used across an inter-process communication and they allow to exchange specific models and numerical algorithms without changes in the source codes. These interfaces are wrapped into the Python scripting language.

The first section of the appendix describes namespaces, packages and classes that are implemented in DIANA. The next three section briefly discuss realization of the DIANA simulation models, dynamic solvers, and parameter continuation solvers. For more detailed information about classes, interfaces and implementation the interested reader can refer to the automatically generated documentation (Pro).

A.1 Class diagrams of DIANA

The general UML package diagram under development is presented in Figure A.1. The diagram depicts the system splitting up into logical groupings in sense of CAPE-OPEN standard interfaces and provides the dependencies among these groupings. The CAPE-OPEN interfaces in two different C++ namespaces are presented (Figure A.1).

The namespace **Common** consists of six packages, namely:

- **Types** is a package for elementary types, interfaces naming, and undefined values.
- **Error** package gives a classification and a hierarchy of potential errors occurring in CAPE-OPEN compliant components.
- **Identification** package declares the identification interface that provides the means for all CAPE-OPEN components to be identified by the name and

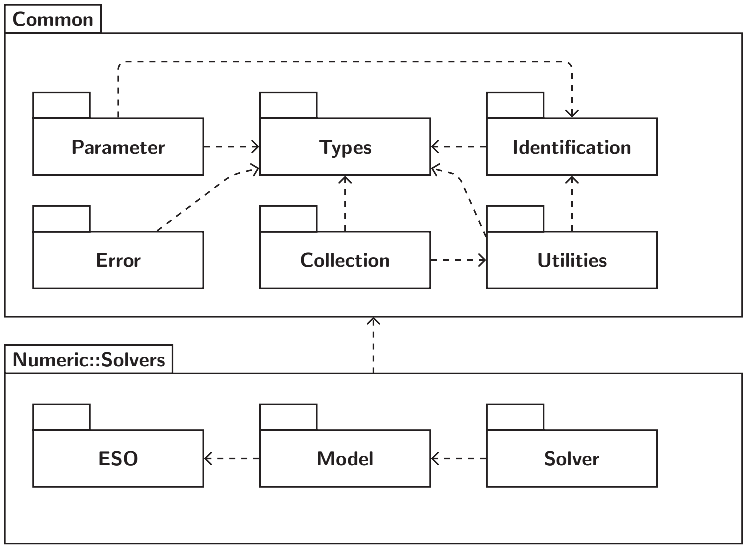


Figure A.1: The package diagram of the implemented CAPE-OPEN interfaces

their textual description.

- **Utilities** package defines the interface that provides the means to set the simulation context, to collect component parameters, to manage a life-cycle of components (creation and termination).
- **Collection** is the package that defines a collection interface. The interface is a standard way to manage the collections of things. The aim of the Collection interface is to give a component the possibility to expose a list of objects to any client of the component.
- **Parameter** package defines a standard access to component parameters. This specification will be used by CAPE-OPEN components wishing to expose some of their internal data to their clients. The interface is made up of two different parts, each corresponding to a different client need: the first part is a fixed, static aspect that describes the parameter, such as a type, name, description, *etc.* The second part deals with the value of the parameter itself.

In the namespace **Numeric::Solvers** interfaces for the dynamic simulation models and solvers are defined. All interfaces in this namespace are derived from the **ICapeIdentification** and **ICapeUtilities** interfaces. The first interface declares identification methods to work with names and descriptions of instances. The second one declares common methods for simulation models and solvers.

A simulation model in the Diana environment is presented as an instance of the **DianaContinuousModel** class (see Figure A.2), which implements interfaces **ICapeNumericModel**, **ICapeNumericContinuousModel** and **IDianaContinuousModel**. The **DianaContinuousModel** class aggregates an instance of an *Equation Set Object* (ESO) that can be retrieved with **GetActiveESO** method. The interface **IDianaContinuousModel** is an extension of the CAPE-OPEN standard and specifies, for example, methods for a generation of linear time-varying variational equations for the initial ESO.

The equation set object in the CAPE-OPEN standard is a software abstraction of a set of nonlinear algebraic (NLAESO) or mixed differential and algebraic equations (DAESO). Dynamical models by the standard interfaces **ICapeNumericESO** and **ICapeNumericDAESO** are presented. This interfaces allow to get an access to specific properties of the system, like state variables, parameters, Jacobian matrices, and so on. Definition of dynamical models in Diana has been presented in Section 2.1. The interface **IDianaDAESO** is an ESO extension and provides methods to work with issues that are not specified in the CAPE-OPEN standard, like higher order Jacobian matrices, absolute tolerances, Jacobian matrices with respect to parameters, and so forth. The implementation of ESO common methods is done in the classes **DianaDAESO** while is responsible for the ESO initialization, access to the ESO properties an instance state save and restore, numerical differentiation with finite differences, and **DianaXmlDAESO**, which is mainly intended for the ESO initialization from an XML file. The class **ModelDAESO** is not included in the **Numeric::Solvers** or **Diana** namespaces and represents a user defined ESO. This class implementation in the C++ language is generated by the ProMoT modeling tool from a user-defined model description (TGZG97).

Definitions of solver interfaces in the namespace **Numeric::Solvers** in Figure A.3 are presented. An **IDASolver** class in the figure is based on the Implicit Differential-Algebraic (IDA) solver from the SUNDIALS library (HBG$^+$05). The class realizes CAPE-OPEN interfaces **ICapeNumericSolver**, **ICapeNumericDAESolver**, that allow to perform basic operation with solvers, like changing tolerances, solution of models, and working with the solution vector. Additional interfaces **IDianaNumericSolver** and **IDianaNumericDAESolver** extend standard interfaces with methods that allow to

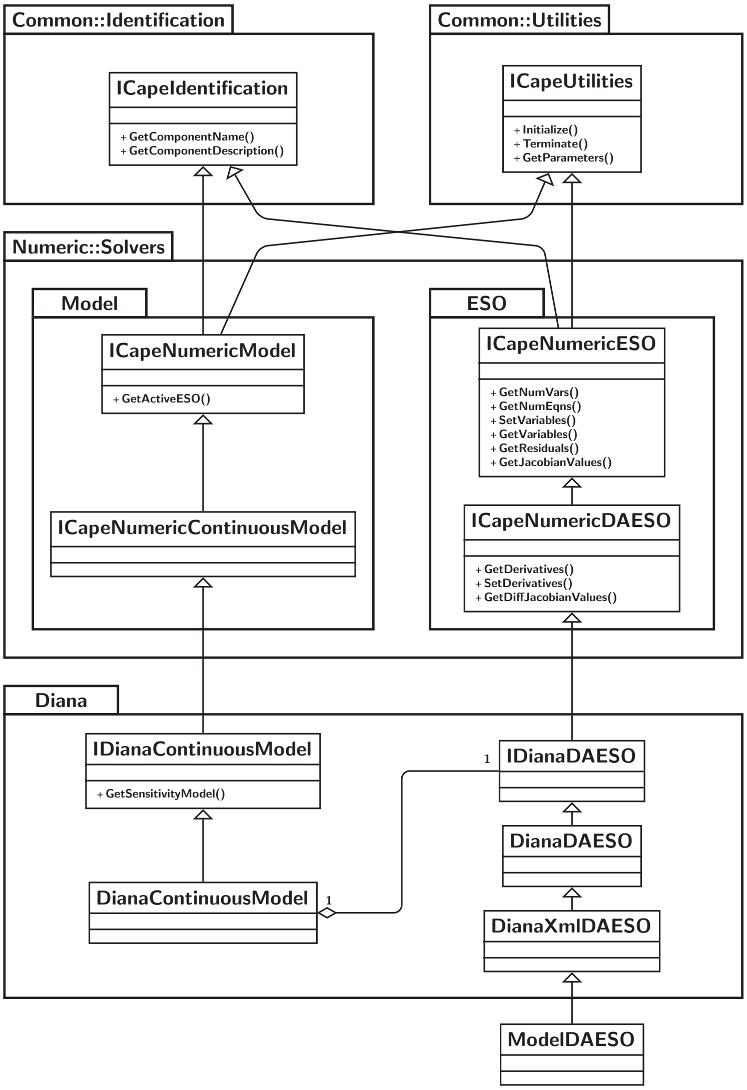


Figure A.2: Diana models class diagram

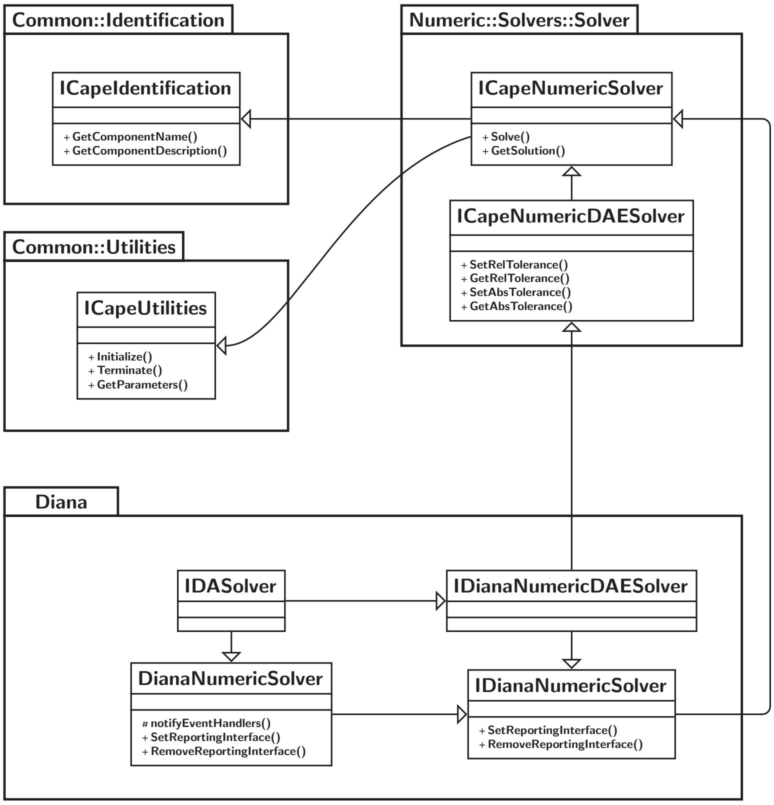


Figure A.3: The IDA solver class diagram

work with results collections or reporting interfaces. The class **IDASolver** realizes a solution of the underlying dynamic system with the help of the IDA solver and will be described in Section A.3.

For the nonlinear analysis various solvers in Diana are implemented. The solvers share the possibility to make a parameter continuation of specific nonlinear phenomena in dynamical systems.

All provided nonlinear solvers are derived from the **DianaContinuation** class which realizes **IDianaContinuation** interface and implements one-parameter pseudo-arc length continuation. The continuation algorithm has been presented in Section 2.2.

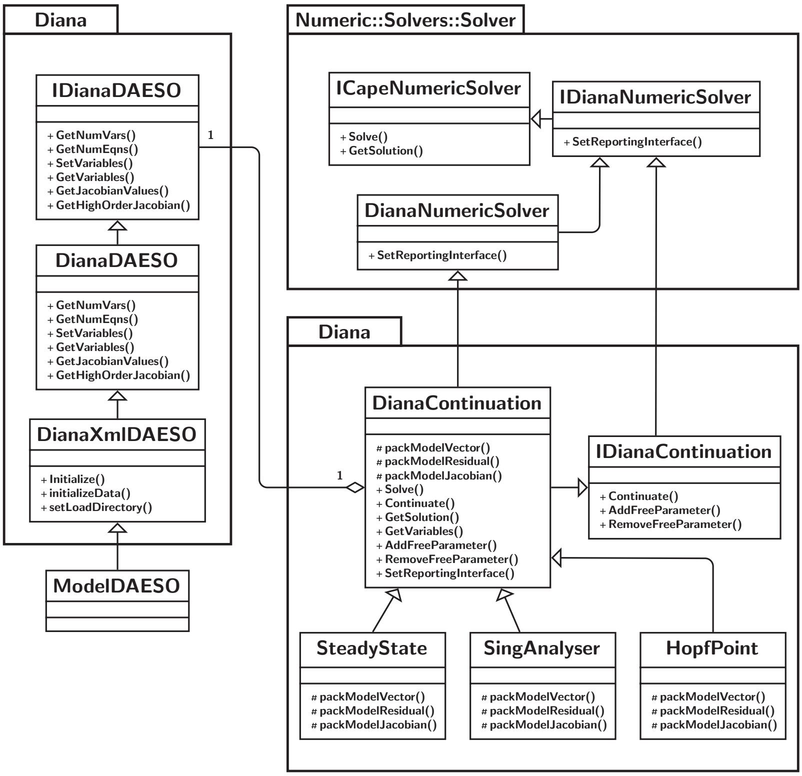


Figure A.4: The Continuation package class diagram

The class **DianaContinuation** has three protected abstract methods **packModelVector**, **packModelResidual**, and **packModelJacobian** that define a nonlinear task. These methods in subclasses **SteadyStateContinuation**, **SingAnalyser** and **HopfPointContinuation** are realized. The first class implements a solver that computes a steady-state solution for the dynamical system. The solver also allows to compute linearized stability criteria for steady states and detects local codim-1 bifurcations. The second solver **SingAnalyser** fulfills model reduction in the presence of a simple zero eigenvalue to a one-dimensional test function. Also the solver computes a set of derivatives of the test function that allows to recognize singularities of the test function solutions curve. The reduction process in Section 2.4 and the solution of the recognition problem in Section 2.5 has been described. Sections 2.3 and 2.7 highlights numerical algorithms and organization of the solver's **SteadyStateContinuation** and **SingAnalyser**.

The creation of model and solver instances is in general performed in a factory design pattern, where the class for the object is identified by name. This abstraction is useful, because it will allow to get the instances in different ways:

(a) as an instance created from a dynamically loaded C++ shared library;

(b) as a direct python instance;

(c) as an remote instance published by an external CAPE enabled simulation package.

The diagrams presented in the section are not complete and only selected classes are presented. The detailed DIANA class diagram can be found in the automatically generated documentation (Pro).

A.2 DIANA simulation models

The simulation model in DIANA presents the continuous-time dynamical system in the DAE numerical form that can be used in numerical computations. The DAE numerical form is described by the component equation set object **DianaDAESO** that encapsulates the differential-algebraic ESO and presents the system of differential-algebraic equations (2.1). The class has the following methods:

- The **Initialize**, **Terminate** methods that are called to initiate and terminate ESO instances.
- The **Get/SetComponentName**, **Get/SetComponentDescription** are access methods to the specified component name and description.
- The **GetNumVars**, **GetNumEqns**, **GetNumRealParams** methods return dimensions of the state variable vector x, residual vector f and parameter vector ν.
- The **GetStateVariables**, **GetParameters** methods return collections with state variables x or parameters ν.
- The **Get/Set(All)Variables** are access methods to the state vector x.
- The **Get/Set(All)Derivatives** are access methods to the derivatives vector $\dot{x}$.

- The **Get/Set(All)Parameters** are access methods to the parameters vector ν.
- The **GetAbsoluteError** method returns an absolute error vector ϵ_{atol} for the state variables.
- The **Get(All)Residuals** methods return the residual vector f.
- The **GetLowerBounds**, **GetUpperBounds** methods return user defined maximal and minimal values for the state variables.
- The **Get/SetIndependentVar** are access methods to the independent variable t.
- The **GetJacobianStruct**. **Get(All)JacobianValues** methods return either the structure or values of the Jacobian matrix $\partial f / \partial x$.
- The **GetDiffJacobianStruct Get(All)DiffJacobianValues** methods return either the structure or values of the Jacobian matrix $\partial f / \partial \dot{x}$.
- The **GetAllParJacobianValues** method returns values of the Jacobian matrix $\partial f / \partial \nu$.
- The **GetHighOrderJacobian** method returns values of the higher order Jacobian matrix $\partial^{(k)} f / \partial \{x, \dot{x}, \nu\}^{(k)}$.
- The **Save/LoadState** are methods that save and restore a state of the ESO instance.
- The **LoadStateDiva** method loads the state variables vector and the parameters vector from Diva generated files.

In addition to the vector ν, the ESO parameters collection also contains the following parameters:

- The **SymbolicJacobian** parameter shows whether a symbolic differentiation (True) or finite differences (False) for the Jacobi matrices computation will be used. If the parameter **SymbolicJacobian** is False, then the following parameters control the calculation of the finite differences:
  - The **FDPartition** shows whether the Minpack DSM algorithm (CGM84) for an optimal or near-optimal consistent partition of the Jacobian matrices will be applied (True) or not (False).
  - The **FDOrder** parameter specifies an approximation order of the finite difference formula.
  - The **FDEpsilon** is the machine-dependent spacing ϵ in equation (2.31).

A.3 Diana dynamic solvers

Dynamic simulation in the Diana environment is performed by a set of numerical differential algebraic solvers. This allows to use the solver instance as a "black-box" integrator in Python scripts. The setup of the solver can be changed by a set of parameters, for example, the relative error tolerance, the type of the internal linear algebra solver, *etc.* This parameter concept is derived from the CAPE-OPEN standard, where parameters have a name and a value, and contain also a specification with a default value and a documentation text, which makes them self-explanatory. Solvers accept different reporting interfaces to process the results of the simulation directly. This allows to generate simulation logs or plots of simulation results. In Diana different integration codes have been used, like BDF or Runge-Kutta, this allows to solve different problem types, like systems of stiff ordinary or differential algebraic equations.

This section describes the **IDASolver** solver (Figure A.3). Information about the other solvers can be found in more recent documentation.

The **IDASolver** class is based on the IDA solver that is a part of the SUNDIALS software family (HBG+05). It solves the initial-value problem (IVP) for the DAE system in the general form (2.1). The integration method used in IDA is the variable-order, and the variable-coefficient BDF (Backward Differentiation Formula) (BCP96). The application of the BDF to the DAE system results in a nonlinear algebraic system to be solved at each step. The solution of the nonlinear system is accomplished with some form of Newton iteration. This leads to a linear system for each Newton correction. The linear systems are solved by the direct dense LAPACK (ABB+99) or sparse UMFPACK (Dav04) linear algebra solvers.

Prior to integrating a DAE initial-value problem, an important requirement is that the pair of vectors x_0 and $\dot{x}_0$ are both initialized to satisfy the DAE residual $f(t_0, x_0, \dot{x}_0, \nu) = 0$. For semi-explicit differential index-one systems, IDA provides a routine that computes consistent initial conditions from a user's initial guess. For this, the solver identifies sub-vectors of x_0, denoted x_d and x_a, which are its differential and algebraic parts, respectively, such that f depends on $\dot{x}_d$ but not on any components of $\dot{x}_a$. The assumption that the system is "index one" means that for a given t and x_d, the system $f(t, x, \dot{x}, \nu) = 0$ defines x_a uniquely. In this case, a solver within IDA computes x_a and $\dot{x}_d|_{t=t_0}$, given x_d and an initial guess for x_a. For the computation of the initial values IDA solves the system $f(t_0, x_0, \dot{x}_0, \nu) = 0$ for the unknown components of x_0 and $\dot{x}_0$, using Newton iteration.

The class **IDASolver** has the following methods:

- The **Initialize**, **Terminate** methods that are called to initiate and terminate solver instances.
- The **Get/SetComponentName**, **Get/SetComponentDescription** are access methods to the specified component name and description.
- The **GetParameters** method returns collections of solver parameters.
- The **Solve** method starts the solution of the ESO that is associated with the solver instance.
- The **GetSolution** method returns the solution vector x.
- The **Get/SetRelTolerance** are access methods to the relative tolerance vector ϵ_{rel}.
- The **Get/SetAbsTolerance** are access methods to the absolute tolerance vector ϵ_{abs}.

Parameters of the solver class are:

- The **Start** parameter specifies the start of a new simulation (True) or continuation of the current one (False).
- The **CalcIC** parameter controls whether consistent initial conditions are computed at the initial time (True) or not (False).
- **T** is the current value of the independent variable t.
- **T0** is the starting value of the independent variable.
- **Tend** is the final value of the independent variable.
- **Tout** is an interval for an equidistant output mode.
- The **Intermediate** parameter with the True value tells the solver to take one internal step and to return the solution at the point reached by that step, otherwise integration proceeds to the parameter value **Tend** without interruption.
- The **VerboseLevel** is a verbosity level of the solver.

- The **MaxInternalSteps** parameter specifies the maximum number of steps to be taken by the solver in its attempt to reach the next output time.
- **LASolver** specifies whether the **dense** or **sparse** linear algebra solver will be used.

A.4 Diana continuation solvers

The continuation algorithm in the class **DianaContinuation** is realized (Figure A.4). The class **DianaContinuation** implements methods of the interfaces **IDianaNumericSolver** and **IDianaContinuation**:

- The **Solve** method solves a nonlinear task $f(x, \lambda) = 0$ for the constant parameter λ.
- The **Continuate** method performs a continuation of the underlying nonlinear task (2.4).
- The **AddFreeParameter** method adds the parameter λ to the nonlinear system (2.4).
- The **RemoveFreeParameter** method removes the parameter λ from the nonlinear system.

Selected parameters of the continuation solver class are:

- The **Parametrization** parameter specifies the parametrization type, either **PseudoArclength** or **Local**.
- The **Predictor** is the predictor type (**Tangent** or **Chord**)
- The **StepSize** is the current step size $\sigma^{(k)}$.
- The **InitialStepSize** is the initial step size $\sigma^{(1)}$.
- The **InitialDirection** is the initial direction of a continuation (possible values are 1 or -1).
- The **SucceededStepScale** is the step size scaling factor ξ_1.
- The **FailedStepScale** is the step size scaling factor ξ_2.
- The **SuccessIts** is a number of the Newton iterations N_1 for the successful step.

- The **FailedIts** is a number of the Newton iterations N_2 for the failed step.
- The **StepNumber** is a number of the continuation step k.
- The **MinStepSize** is the minimal step size σ_{min}.
- The **MaxStepSize** The is the maximal step size σ_{max}.
- The **MaxStepsAmount** is the maximal number of steps k_{max}.
- The **MaxNewtonIts** is the maximal number of Newton iterations N_{max}.

A.4.1 Steady-state continuation solver

The class **SteadyStateContinuation** (see Figure A.4) inherits the parameter continuation algorithm from **DianaContinuation** and implements the protected methods **packModelVector**, **packModelResidual**, and **packModelJacobian**. The methods assemble the steady state continuation problem in form (2.3) that is based on the model ESO (2.1). The class **SteadyStateContinuation** extends the parameter set of the base solver with the following parameters:

- The **EigSolver** parameter specifies the algorithm for the eigenvalues computation. The value of the parameter may be either **Lapack** or **Arpack**.
- The **EigMonitor** parameter shows a number of monitored eigenvalues (0¡**EigMonitor**$\leqslant n$)
- The **EigConverged** is a number of converged eigenvalues.
- The **Stability** shows the local stability of the computed steady-state point.
- The **StabilityCheck** parameter controls whether the linearized stability will be computed (True) or not (False).
- The **ConditionCheck** parameter specifies the bifurcation type that will be checked during the continuation. Possible types are: a real eigenvalue crossing the imaginary axis (**SteadyStateZRE**), a pair of complex conjugate eigenvalues crossing the imaginary axis (**SteadyStateZCE**), and the **SteadyStateNone** value that specifies an empty condition. The test condition can be combined with a bitwise OR operator —, for example, an expression **SteadyStateZRE**—**SteadyStateZCE** specifies monitoring of both conditions.

A.4.2 Singularity analyzer solver

The class **SingAnalyser** implements the augmented system (2.33) and the test functions (2.34) in the abstract methods **packModelVector**, **packModelResidual** and **packModelJacobian** of the base **DianaContinuation** class (see Figure A.4). In the specified nonlinear system the unknown variables vector has $3n + 2 + k + 1$ variables and consists of the model state variables x, the eigenvectors v_0 and v_0^*, the singular values β and γ, and $k + 1$ model parameters α. The residuals vector has $3n + 2$ augmented system equations, k test functions, and the parametrization equation.

The class **SingAnalyser** extends the parameter set of the base solver with the following parameters:

- The **ConditionEquations** parameter determines a set of the test functions that will be added to the augmented system (2.33). Possible values are:

SingularityNone	empty set of the test functions,
SingularityGx	$g_z = 0$,
SingularityGxx	$g_{zz} = 0$,
SingularityGp	$g_\lambda = 0$,
SingularityGxp	$g_{z\lambda} = 0$.

  The test functions can be combined with a bitwise OR operator —, for example, an expression **SingularityGx**—**SingularityGxx** specifies $g_z = g_{zz} = 0$.

- The **ConditionCheck** parameter indicates a set of the test functions that will be checked for zero-crossing values during the continuation. If one of the test values crosses zero line, the solver stops the continuation and returns the **ContiOkTestFunction** return value.

- The **ConditionShow** parameter describes a set of the test functions that will be computed but not checked during the continuation.

- The **ConditionCurrent** parameter contains an identifier of the test function that had changed the sign if the solver returned **ContiOkTestFunction**.

- The **LambdaParameter** parameter specifies the model name of the parameter λ under consideration of the recognition process.

- **Gx**, **Gxx**, **Gp**, **Gxp** are values of the test functions g_z, g_{zz}, g_λ, and $g_{z\lambda}$, respectively, for the calculated continuation point.

The initial state variables and parameters values for the singularity analysis solver are taken from the steady-state point continuation that returns the **SteadyStateZRE** condition flag or from a previous singular points variety continuation. An initial approximation of the values β and γ and corresponding left and right eigenvectors v_0 and v_0^* are computed from the eigenvalue problems

$$\begin{aligned} f_x^T f_x v_0 &= \beta^2 v_0, \\ f_x f_x^T v_0^* &= \gamma^2 v_0^*, \end{aligned}$$

at the starting continuation point with the ARPACK library function `naupd`.

A.4.3 Hopf point continuation solver

The class **HopfPointContinuation** implements the augmented system (2.36) in the abstract methods **packModelVector**, **packModelResidual**, and **packModelJacobian** of the base class **DianaContinuation** (see Figure A.4). In the specified nonlinear system the unknowns vector has $3n+3$ variables and consists of the model state variables x, the real u and imaginary w parts of the eigenvector, the imaginary part of eigenvalue ω_0, and the two model parameters from the vector ν. The residual vector has $3n+2$ augmented system equations (2.36) and the parametrization equation (see 2.2.2).

The class **HopfPointContinuation** extends the parameter set of the base solver with the following parameters:

- The **EigMonitor** parameter shows a number monitored eigenvalues (0¡**EigMonitor**$\leqslant n$)
- The **EigConverged** is a number of converged eigenvalues.

The initial state variables and parameters values for the Hopf point continuation are taken from the steady-state point continuation that returns **SteadyStateZCE** condition flag. An initial approximation of the eigenvalue $i\omega_0$ and corresponding eigenvector $u_i w$ is computed from the generalized eigenvalue problem (2.37) at the initial point with LAPACK or ARPACK methods as in the **SteadyStateContinuation** class.

Bibliography

[ABB$^+$99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, third ed., SIAM, Englewood Cliffs, NJ, 1999.

[AMR95] U. Ascher, R. Mattheij, and R. Russell, *Numerical Solution of Boundary Value Problems*, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1995.

[Arn83] V. Arnold, *Geometrical methods in the theory of ordinary differential equations*, Springer-Verlag, New York, 1983.

[BBC$^+$94] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2 ed., SIAM, Philadelphia, PA, 1994.

[BCP96] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Englewood Cliffs, NJ, 1996.

[BG94] J. Barrow-Green, *Oscar II's prize competition and the error in Poincaré's memoir on the three body problem*, Archive for History of Exact Sciences **48** (1994), no. 2, 107—-131.

[CGM84] T. F. Coleman, B. S. Garbow, and J. J. More, *Software for estimating sparse Jacobian matrices*, ACM Trans. Math. Software **10** (1984), 329–345.

[D$^+$02] E. J. Doedel et al., *AUTO 2000: Continuation and Bifurcation Software for Ordinary Differential Equations*, Tech. report, Caltech, 2002.

[Dav04] T. A. Davis, *Algorithm 832: UMFPACK — an unsymmetric-pattern multifrontal method with a column pre-ordering strategy*, ACM Transactions on Mathematical Software **30** (2004), 196–199.

[DGK$^+$06] A. Dhooge, W. Govaerts, Y. A. Kuznetsov, W. Mestrom, A. Riet, and B. Sautois, *MATCONT and CL_MATCONT: Continuation toolboxes in Matlab*, Tech. report, Utrecht University, The Netherlands, December 2006.

[dHR81] C. den Heijer and W. C. Rheinboldt, *On Steplength Algorithms for a Class of Continuation Methods*, SIAM Journal on Numerical Analysis **18** (1981), 925–948.

[DKKvV07] E. Doedel, B. Kooi, Y. Kuznetsov, and G. van Voorn, *Continuation of connecting orbits in 3D-ODEs: (I) Point-to-cycle connections*, Submitted for publication, 2007.

[Don02a] J. Dongarra, *Basic Linear Algebra Subprograms Technical Forum Standard*, International Journal of High Performance Applications and Supercomputing **16** (2002), no. 1, 1–111.

[Don02b] J. Dongarra, *Basic Linear Algebra Subprograms Technical Forum Standard*, International Journal of High Performance Applications and Supercomputing **16** (2002), no. 2, 115–199.

[DR04] L. Dieci and J. Rebaza, *Point-to-Periodic and Periodic-to-Periodic Connections*, BIT Numerical Mathematics **44** (2004), no. 1, 41–62.

[Erm02] B. Ermentrout, *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*, SIAM, Philadelphia, 2002.

[Far94] M. Farkas, *Periodic motions*, Applied Mathematical Sciences, no. 104, Springer-Verlag, New York, 1994.

[FdSMY04] R. J. Fateman, P. N. de Souza, J. Moses, and C. Yapp, *The Maxima Book*, 2004, URL `http://maxima.sourceforge.net`.

[Gea88] C. W. Gear, *Differential-algebraic equations index transformations*, SIAM Journal on Scientific and Statistical Computing **9** (1988), no. 1, 39–47.

[Gea90] C. W. Gear, *Differential Algebraic Equations, Indices, and Integral Algebraic Equations*, SIAM Journal on Numerical Analysis **27** (1990), no. 6, 1527–1534.

[GKN$^+$03] M. Ginkel, A. Kremling, T. Nutsch, R. Rehner, and E. Gilles, *Modular modeling of cellular systems with ProMoT/Diva*, Bioinformatics **19** (2003), 1169–1176.

[GKS98] W. Govaerts, Y. A. Kuznetsov, and B. Sijnave, *Implementation of Hopf and double-Hopf continuation using bordering methods*, ACM Trans. Math. Softw. **24** (1998), no. 4, 418–436.

[GM97] V. Gehrke and W. Marquardt, *A Singularity Theory Approach to the Study of Reactive Distillation*, Computers and Chemical Engineering **21** (1997), S1001–S1006.

[GMM05] J. Gerhard, M. Mönnigmann, and W. Marquardt, *Constructive Nonlinear Dynamics –– Foundations and Application to Robust Nonlinear Control*, Control and Observer Design for Nonlinear Finite- and Infinite-Dimensional Systems (T. Meurer, K. Graichen, and E.-D. Gilles, eds.), Springer-Verlag, 2005, pp. 165–182.

[Gro59] D. Grobman, *Homeomorphisms of systems of differential equations*, Dokl. Akad. Nauk SSSR **128** (1959), 880–881.

[GS85] M. Golubitsky and D. G. Schaeffer, *Singularities and groups in bifurcation theory. Vol. I*, Springer-Verlag, New York, 1985.

[GV96] G. H. Golub and C. F. Van Loan, *Matrix Computations*, third ed., The John Hopkins University Press, Baltimore and London, Baltimore, 1996.

[Haf68] C. Hafke, *Experimentelle Untersuchungen des dynamischen Verhaltens von Rührkesselreaktoren.*, Messen, Steuern, Regeln **11** (1968), 204–208.

[Har60] P. Hartman, *A Lemma in the Theory of Structural Stability of Differential Equations*, Proceedings of the American Mathematical Society, **11** (1960), no. 4, 610–620.

[HBG$^+$05] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, *SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers*, ACM Transactions on Mathematical Software **31** (2005), no. 3, 363–396.

[Hen03] M. E. Henderson, *Multiple Parameter Continuation: Computing Implicitly Defined k-manifolds*, International Journal of Bifurcation and Chaos **12** (2003), no. 3, 451–476.

[Hig96] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[HK91] J. K. Hale and H. Koçak, *Dynamics and bifurcations*, Springer-Verlag, New York, 1991.

[HKW81] B. D. Hassard, N. D. Kazarinoff, and Y.-H. Wan, *Theory and Applications of Hopf Bifurcation*, Cambridge University Press, Cambridge, MA, 1981.

[HW96] E. Hairer and G. Wanner, *Solving ordinary differential equations II: Stiff and differential-algebraic problems*, second ed., Springer-Verlag, 1996.

[JKB$^+$99] M. Jarke, J. Köller, B. Braunschweig, W. Marquardt, and L. von Wedel, *CAPE-OPEN: Experiences from a Standardization Effort in Chemical Industries*, Proc. of 1st IEEE Conference on Standardisation and Innovation in Information Technology (SIIT 99) (Aachen, Germany), 1999, pp. 25–35.

[JM82] H. Jarausch and W. Mackens, *CNSP — A Fast, Globally Convergent Scheme to Compute Stationary Points of Elliptic Variational Problems*, Tech. report, Institut für Geometrie und Praktische Mathematik, Aachen, Germany, 1982.

[JM84] H. Jarausch and W. Mackens, *Numerical treatment of bifurcation branches by adaptive condensation*, Numerical Methods for Bifurcation Problems **70** (1984), 296–309.

[JM87] H. Jarausch and W. Mackens, *Computing Bifurcation Diagrams for Large Nonlinear Variational Problems*, Progress in Large Scale Scientific Computing (Basel) (P. Deuflhard and B. Engquist, eds.), vol. 7, Birkhauser Verlag, 1987.

[Kel77] H. B. Keller, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, Applications of Bifurcation Theory (New York) (P. Rabinowitz, ed.), Academic Press, 1977, pp. 159–384.

[KKLN93] A. I. Khibnik, Y. A. Kuznetsov, V. V. Levitin, and E. V. Nikolaev, *Continuation techniques and interactive software for bifurcation analysis of ODEs and iterated maps: physics*, Proceedings of a NATO advanced research workshop held at the Centre for Nonlinear Phenomena and Complex Systems on Homoclinic Chaos (1993), 360–371.

[KL97] Y. A. Kuznetsov and V. V. Levitin, *CONTENT: A multiplatform environment for continuation and bifurcation analysis of dynamical systems*, Tech. report, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, 1997.

[Kub76] M. Kubíček, *Algorithm 502: Dependence of Solution of Nonlinear Systems on a Parameter*, ACM Transactions on Mathematical Software **2** (1976), no. 1, 98–107.

[Kun91] P. Kunkel, *Augmented systems for generalised turning points*, Bifurcation and Chaos: Analysis, Algorithms, Applications (Basel) (R. Seydel, ed.), vol. 97, Birkhäuser Verlag, 1991, pp. 231–236.

[Kuz04] Y. A. Kuznetsov, *Elements of applied bifurcation theory*, third ed., Applied Mathematical Sciences, vol. 112, Springer-Verlag, New York, 2004.

[LMW98] R. Lamour, R. März, and R. Winkler, *How Floquet-theory Applies to Index 1 Differential-Algebraic Equations*, Journal of Mathematical Analysis and Applications **217** (1998), 372–394.

[LP99] S. Li and L. Petzold, *Design of new daspk for sensitivity analysis*, Tech. report, University of California, Santa Barbara, 1999.

[LP00] S. Li and L. R. Petzold, *Software and Algorithms for Sensitivity Analysis of Large-Scale Differential Algebraic Systems*, J. Comput and Appl. Math (2000), 131–145.

[LRSC98] K. Lust, D. Roose, A. Spence, and A. Champneys, *An Adaptive Newton-Picard Algorithm with Subspace Iteration for Computing Periodic*

Solutions, SIAM Journal on Scientific Computing **19** (1998), no. 4, 1188–1209.

[LSY98] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Englewood Cliffs, NJ, 1998.

[Lus97] K. Lust, *Numerical bifurcation analysis of periodic solutions of partial differential equations*, Ph.D. thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1997.

[MKG$^+$99] M. Mangold, A. Kienle, E.-D. Gilles, M. Richter, and E. Roschka, *Coupled reaction zones in a circulation loop reactor*, Chemical Engineering Science **54** (1999), 2597–2607.

[MKMG00] M. Mangold, A. Kienle, K. D. Mohl, and E. D. Gilles, *Nonlinear computation in DIVA — methods and applications*, Chemical Engineering Science (2000), 441–454.

[MKS04] M. Mangold, M. Krasnyk, and K. Sundmacher, *Nonlinear Analysis of Current Instabilities in High Temperature Fuel Cells*, Chemical Engineering Science **59** (2004), no. 22–23, 4869–4877.

[MKS06] M. Mangold, M. Krasnyk, and K. Sundmacher, *Theoretical Investigation of Steady State Multiplicities in Solid Oxide Fuel Cells*, Journal of Applied Electrochemistry **36** (2006), no. 3, 265–275.

[MS93] S. E. Mattsson and G. Söderlind, *Index Reduction in Differential-Algebraic Equations Using Dummy Derivatives*, SIAM Journal on Scientific Computing **14** (1993), no. 3, 677–692.

[NW91] U. Nowak and L. Weimann, *A Family of Newton Codes for Systems of Highly Nonlinear Equations*, Tech. report, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, December 1991.

[Pan88] C. C. Pantelides, *The consistent intialization of differential-algebraic systems*, SIAM Journal on Scientific and Statistical Computing **9** (1988), no. 2, 213–231.

[PK02] P. K. Pathath and A. Kienle, *A numerical bifurcation analysis of nonlinear oscillations in crystallization processes*, Chemical Engineering Science **57** (2002), no. 10, 4391—4399.

[Pro] *ProMoT/Diana documentation*, Max-Planck-Institute, Magdeburg, URL `http://promottrac.mpi-magdeburg.mpg.de/doc/`.

[PS82] C. C. Paige and M. A. Saunders, *LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares*, ACM Transactions on Mathematical Software **8** (1982), no. 1, 43–71.

[PyD] *PyDSTool Wiki*, Cornell University, URL `http://www.cam.cornell.edu/~rclewley/cgi-bin/moin.cgi/`.

[Rei91] S. Reich, *On an existence and uniqueness theory for nonlinear differential-algebraic equations*, Source Circuits, Systems, and Signal Processing **10** (1991), no. 3, 343–359.

[RL88] A. D. Randolph and M. A. Larson, *Theory of Particulate Processes*, Academic Press, Inc., 1988.

[RMK$^+$06] R. Radichkov, T. Müller, A. Kienle, S. Heinrich, M. Peglow, and L. Mörl, *A numerical bifurcation analysis of continuous fluidized bed spray granulation with external product classification*, Chemical Engineering and Processing **45** (2006), no. 10, 826–837.

[S$^+$02] A. G. Salinger et al., *Library Of Continuation Algorithms: Theory and Implementation Manual*, 2002.

[Saa92] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, 1992.

[Saa00] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2000.

[Sey93] R. Seydel, *BIFPACK–– A program package for continuation, bifurcation and stability analysis. Version 2.4*, Tech. report, Mathematische Institute der Julius-Maximilians-Universität, Würzburg, Germany, 1993.

[Sey94] R. Seydel, *Practical bifurcation and stability analysis: from equilibrium to chaos*, Springer-Verlag, New York, 1994.

[SK93] G. M. Shroff and H. B. Keller, *Stabilization of Unstable Procedures: The Recursive Projection Method*, SIAM Journal on Numerical Analysis **30** (1993), no. 4, 1099–1120.

[SRP$^+$98] R. Schütte, A. Ruhs, I. Pelgrims, C.-J. Klasen, and L. Kaiser, *Verfahren zur Herstellung von Granulat mit periodisch oszillierender Korngrößenverteilung und Vorrichtung zu seiner Durchführung*, 1998, Patent Application: DE 19639579 C1.

[TGZG97] F. Tränkle, A. Gerstlauer, M. Zeitz, and E. D. Gilles, *The object-oriented model definition language MDL of the knowledge-based process modeling tool ProMoT*, Proc. 15th IMACS World Congress (Berlin, Germany) (A. Sydow, ed.), vol. 3, Wissenschaft und Technik Verlag, 1997, pp. 91–96.

[TZGG00] F. Tränkle, M. Zeitz, M. Ginkel, and E. D. Gilles, *ProMoT: A Modeling Tool for Chemical Processes*, Mathematical and Computer Modelling of Dynamical Systems **6** (2000), no. 3, 283–307.

[vH58] C. van Heerden, *The character of the stationary state of exothermic processes*, Chemical Engineering Science **7** (1958), 133–145.

[Wan94] D. Wang, *Differentiation and Integration of Indefinite Summations with Respect to Indexed Variables — Some Rules and Applications*, Journal of Symbolic Computation (1994), 249–263.

[WAPGK06] R. Waschler, O. Angeles-Palacios, M. Ginkel, and A. Kienle, *Object-oriented modeling of large-scale chemical engineering processes with ProMoT*, Mathematical and Computer Modelling of Dynamical Systems **12** (2006), no. 1, 5–18.

[YRSM$^+$04] Y. Ye, L. Rihko-Struckmann, B. Munder, H. Rau, and K. Sundmacher, *Feasibility of an Electrochemical Membrane Reactor for Partial Oxidation of n-Butane to Maleic Anhydride*, Ind. Eng. Chem. Res. **43** (2004), 4551–4558.

[ZMOG99] K. P. Zeyer, M. Mangold, T. Obertopp, and E. D. Gilles, *The iron(III)-catalyzed oxidation of ethanol by hydrogen peroxide: a thermokinetic oscillator*, Journal of Physical Chemistry **103A** (1999), 5515–5522.

[ZPMG00] K. P. Zeyer, S. Pushpavanam, M. Mangold, and E. D. Gilles, *The behavior of the iron(III)-catalyzed oxidation of ethanol by hydrogen peroxide in a fed-batch reactor*, Physical Chemistry, Chemical Physics **2** (2000), 3605–3612.

Index

bifurcation, 9, 26
  Bogdanov-Takens, 39
  double-Hopf, 40
  fold-Hopf, 39
  generalized Hopf, 40
  Hopf point, 22
  limit point, 22
bifurcation analysis, 9

characteristic
  exponent, 50
characteristic multiplier, 49
codimension, 28
continuation algorithm, 19
critical point, 22

differential equation, 11
differential index, 12
differential-algebraic equation, 11
dynamical system, 11

equation set object, 109
evaluation operator, 11

heteroclinic orbit, 40
homoclinic orbit, 39
hyperbolic equilibrium, 21

initial state, 11

limit point, 14

monodromy matrix, 48

normal form, 27

organizing center, 29

parametrization, 14
  local, 16
  pseudo-arc length, 17
periodic orbit, 47
phase condition, 47
Poincaré map, 50
predictor
  chord, 16
  tangent, 15

recognition problem, 27
regular solution, 14

simple fold, 14
simulation model, 13
singularity, 26
state space, 11
step size, 14

test function, 20

universal unfolding, 28

versal unfolding, 27