NTNU
Norwegian University of
Science and Technology

Continuation and Bifurcation software in MATLAB

Eirik Ravnås

Master of Science in Physics and Mathematics
Submission date: August 2008
Supervisor: Anne Kværnø, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Problem Description

Implement a continuation and bifurcation software in MATLAB with the aim of being sufficently fast and robust to obtain the solution branches of periodic solutions originating from the study of moving mesh partial differential equations described in the project work.

Assignment given: 11. March 2008
Supervisor: Anne Kværnø, MATH

Continuation and Bifurcation Software in MATLAB

Eirik Ravnås

Abstract

This article contains discussions of the algorithms used for the construction of the continuation software implemented in this thesis. The aim of the continuation was to be able to perform continuation of equilibria and periodic solutions originating from a Hopf bifurcation point. Algorithms for detection of simple branch points, folds, and Hopf bifurcation points have also been implemented. Some considerations are made with regard to optimization, and two schemes for mesh adaptation of periodic solutions based on moving mesh equations are suggested.

Preface

This Master's thesis completes my five year Master of Science program. It has been carried out at the Department of Mathematical Sciences of the Norwegian University of Science and Technology (NTNU) during the spring and summer of 2008 under supervision of Anne Kværnø. The thesis origins the project work carried out during the autumn 2007, where the continuation toolbox CL_MATCONT was used for analyzing the dynamical behavior of some moving mesh partial differential equations (MMPDEs) when applied to a known solution function. CL_MATCONT performed well when it came to computing solution branches of equilbria as well as for detecting folds and Hopf bifurcation points along the branch, but the continuation of periodic solutions were somewhat frustrating since despite all my attempts I could not CL_MATCONT to compute satisfactory branches of periodic solutions. Sometimes the continuation failed already at the initial point, and sometimes a little further out in the solution, but never far enough as to provide some useful insight of the dynamic behaviour of the MMPDEs. All this sparked the interest of making a continuation software, as to understand the mechanisms involved, and with the goal of making a program sufficiently fast and robust as to compute the branches of periodic solutions for the test examples described in the project work [**14**].

I would like to give thanks to my supervisor Anne Kværnø for suggesting this interesting topic for my thesis and for her help and comments. Credit should also be given to Greg von Winckel for having made available a matlab file for calculating gauss points on an arbitrary interval at the Matlab file exhange, and to the makers of MATCONT. Peeking at their code has provided assistance at the darkest of times... In particular, I have made use of their table for the Newton-Cotes quadrature weights. Finally, I would like to give my heartiest thanks to Cecilie and our son Leander for their patience during these past months.

Stavanger, 12.08.2008 Eirik Ravnås

Contents

CHAPTER 1

Introduction

Numerical continuation as a tool for analyzing nonlinear equations has proven very useful especially in the past decades as computers and algorithms has constantly involved. Even though the use of continuation techniques can be traced back to works of Poincarè(1881-1886), Klein (1882-1883) and Bernstein (1910) [**6**], and the use of deformations to solve nonlinear equations may be traced back at least to Lahaye (1934), the golden age for these methods began in the late 1970's. Perhaps to some degree triggered by Keller's pseudo-arclength continuation technique (1977), and implemented by Eusebius Doedel in the program AUTO (1980).

There are two main branches for continuation software, namely those based on predictor-corrector (PC) methods, and those based by piecewise linear continuation techniques (PL). In general, it may be stated that PC-methods have performed best when high accuracy is needed, while PL-methods are perhaps better suited for a more qualitative global study. There have been developed several software packages for continuation. Most notably is perhaps, AUTO, MATCONT (or CL_MATCONT) (Govaerts et.al) which is a successor of CONTENT (Kuznetsov et.al), LOCA (Sandia Corp.) and PDDE-CONT (Szalai). Wikipedia's list of continuation software packages currently contains 13 entries, whereof many are open source projects.

There may not be an immediate need for adding yet another program to this list, as we have not tested other programs than CL_MATCONT for the specific test examples that motivated this work, but it is an interesting topic, and it also provides much insight to the mechanisms and particular problems that needs to be addressed.

The disposition of this thesis is as follows; in the second chapter we shall briefly discuss the concept of numerical continuation and review basic terminology important results of the subject of dynamical systems, and in particular those originating from autonomous systems of ordinary differential equations. In the third chapter we introduce PC methods in general, and in the fourth chapter we formulate the systems for continuation of equilibria and periodic solutions, as well discuss methods of detecting folds, simple branch points and Hopf bifurcations along solution branches of equilibria. Chapter 5 addresses topics regarding optimization and implementation that were not explained elsewhere, and a brief user manual is contained in chapter 6. In chapter 7 the software is tested on some numerical examples.

CHAPTER 2

Preliminaries

This chapter aims at presenting the basic theorems, notation and terminology that will be used in the rest of this text. The reader is assumed to be familiar with elementary linear algebra and dynamical systems at an introductory level. The reader may consult [**10**],[**9**], [**6**] and [**2**] for a more detailed account of the preliminaries.

1. Notation

We shall employ the following notation throughout this paper:

$\mathbb{R}$	denotes the set of real numbers
$\mathbb{Z}$	denotes the set of integers
id	denotes the identity map
A^*	means the complex conjugate of a matrix or a vector A
I_n	denotes the $n \times n$ identity matrix
$0_{n\times m}$	denotes the $n \times m$ zero matrix
$\mathcal{O}$	denotes the Landau big-O symbol

2. Numerical Continuation

We shall consider equations of the form

$$f(x) = 0, \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$. We shall occasionally write x in decomposed form;

$$x = (u, \lambda), \tag{2}$$

where $u \in \mathbb{R}^n$ can be though of as a variable and $\lambda \in \mathbb{R}$ as a parameter. Although this distinction is not necessary, it adds meaning since we can regard (1) as a dynamical system. Suppose that x_0 is a root of f, the purpose of the analysis here is then to investigate how the the set $f^{-1}(0)$ changes locally about $x_0 \equiv (u_0, \lambda_0)$as the parameter λ change. Let U be a sufficiently small neighborhood of x_0, then from the *implicit function theorem* 1 it can be found certain conditions for which the set $f^{-1}(0) \cap U$ is a curve. The purpose of numerical continuation is to trace out this curve.

THEOREM 1 (Implicit function theorem). *Let f be as above* (1) *and suppose*

- $f(u_0, \lambda_0) = 0$,

- *$f_u(u_0, \lambda_0)$ is nonsingular*
- *f and f_u are smooth near u_0*

Then there is a neighborhood, U, of λ_0 such that for $\lambda \in U$, there is a unique smooth function $u(\lambda)$ satisfying

$$f(u(\lambda), \lambda) = 0 \tag{3}$$

and

$$u(\lambda_0) = u_0. \tag{4}$$

Moreover, if $f \in C^m$, then $u(\lambda) \in C^m$.

REMARK 2. *In this paper the smoothness conditions are always assumed to hold.*

The curve $u(\lambda)$ of theorem 1 is called a *solution branch* of f.

DEFINITION 3. *A point x is called a* regular point *of f if the Jacobian f_x has maximal rank. A value $y \in R^n$ is called a* regular value *of f if x is a regular point of f for all $x \in f^{-1}(y)$. Points and values are called* singular *if they are not regular.*

The following reformulation of the implicit function theorem 1 provides the foundation for numerical continuation.

THEOREM 4. *Let $x_0 = (u_0, \lambda_0)$ be a regular solution of $f(x) = 0$, $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$. Then, near x_0, there exists a unique one-dimensional continuum of solutions $x(s)$, called a* solution branch*, with $x(0) = x_0$.*

The proof of this is basically based on interchanging the columns of the Jacobian, and using the implicit function theorem. A complete proof can be found in [**3**]. It is also evident from the implicit function theorem that the solution branch can be parameterized locally by some element of the vector x. It is clear from the theorem 4 that each regular point in $\mathbb{R}^{n+1}$ has a neighborhood in which every point is regular. We may therefore suspect that most points in $\mathbb{R}^{n+1}$ are regular. The proper justification for this suspicion is due to Sard:

THEOREM 5 (Sard's Theorem). *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be C^k, k times continuously differentiable, where $k > \max\{n - m + 1, 1\}$. Let X be the critical set of f, i.e the set of points $x \in \mathbb{R}^n$ at which the Jacobian matrix of f has rank $< m$. Then $f(X)$ has Lebesgue measure 0 in $\mathbb{R}^m$.*

By differentiating

$$f(x(s)) = 0$$

with respect to s, we obtain

$$f_x(x(s))x'(s) = 0.$$

Thus

$$x'(s) \in \ker f_x,$$

and the solution branch proceeds in the direction of a null vector of f_x. For a regular point the null vector is unique up to scalar multiplication. Note that this makes sense since $x(s)$ is a contour curve in $\mathbb{R}^{n+1}$. This *direction vector* $x'(s)$ gives us an orientation of the curve, with respect to increasing parameter values s, and it is clear that this orientation depends on the parameterization of the curve. Another commonly used orientation of the curve that do not depend on the parameterization is the tangent:

DEFINITION 6. *Let A be an $n \times (n+1)$-matrix with* $\operatorname{rank}\{A\} = n$*. The unique vector $t(A) \in \mathbb{R}^{n+1}$ satisfying the three conditions*

(1) $At = 0$*;*

(2) $\| t \| = 1$*;*

(3) $\det \begin{pmatrix} A \\ t^* \end{pmatrix} > 0$*;*

is called the tangent vector induced by A.

LEMMA 7. *The set $\mathcal{M}$ of all $n \times (n+1)$-matrices A having maximal rank n is an open subset of $\mathbb{R}^{n\times(n+1)}$, and the map $A \in \mathcal{M} \mapsto t(A)$ is smooth.*

PROOF. [**6**] □

PROPOSITION 8. *The function $k : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ defined by $k = f_x \circ t$ is smooth in a neighborhood of a regular point x.*

PROOF. The Jacobian function f_x is already assumed to be smooth in a neighborhood of x, and since t also is smooth the composition k must also be smooth in a neighborhood of x. □

Note that $\begin{pmatrix} A \\ t \end{pmatrix}$ can only be zero if the Jacobian A is singular, hence if we consistently trace the curve $x(s)$ in the direction of the tangent, we are ensured to trace the curve in one direction. Note that this argument is no longer valid if we cross singular points.

3. Dynamical Systems

The purpose of this section is to establish the terminology and basic results that are used in this paper. Most of the definitions and theorem in the first part of this section can be found in [**9**].

DEFINITION 9. *A* dynamical system *is a triple $\{\mathcal{T}, \mathcal{B}, \phi^t\}$, where $\mathcal{T}$ is a time set, $\mathcal{B}$ is a state space, and $\phi^t : \mathcal{B} \to \mathcal{B}$ is a family of evolution operators parameterized by $t \in \mathcal{T}$ and satisfying the properties*

(1) $\phi^0 = id$,

(2) $\phi^{t+s} = \phi^t \circ \phi^s$.

Note that the state space $\mathcal{B}$ in the context of this paper can be thought of as a Banach space.

DEFINITION 10. *An orbit starting at u_0 is an ordered subset of the state space $\mathcal{B}$,*

$$\mathrm{Or}(u_0) = \{u \in \mathcal{B} : u = \phi^t u_0, \forall t \in \mathcal{T} \text{ such that } \phi^t u_0 \text{ is defined}\}.$$

DEFINITION 11. *The* phase portrait *of a dynamical system is a partitioning of the state space into orbits.*

DEFINITION 12. *A point $u_0 \in \mathcal{B}$ is called an equilibrium point (fixed point) if $\phi^t u_0 = u_0$ for all $t \in \mathcal{T}$.*

DEFINITION 13. *A cycle is a periodic orbit, namely a non-equilibrium orbit L_0, such that each point $u_0 \in L_0$ satisfies $\phi^{t+T_0} u_0 = \phi^t u_0$ with some $T_0 > 0$, for all $t \in \mathcal{T}$. The minimal T_0 with this property is called the* period *of the cycle L_0.*

DEFINITION 14. *A cycle of a continuous-time dynamical system, in a neighborhood of which there are no other cycles, is called a* limit cycle.

DEFINITION 15. *An invariant set of a dynamical system $\{\mathcal{T}, \mathcal{B}, \phi^t\}$ is a subset $S \subset \mathcal{B}$ such that $u_0 \in S$ implies $\phi^t u_0 \in S$ for all $t \in \mathcal{T}$.*

THEOREM 16. *An invariant set S_0 is stable if the two following conditions hold;*

(1) *for any sufficiently small neighborhood $U \supset S_0$ there exists a neighborhood $V \supset S_0$ such that $\phi^t u \in U$ for all $u \in V$ and all $t > 0$;*

(2) *there exists a neighborhood $U_0 \supset S_0$ such that $\phi^t u \to S_0$ for all $u \in U_0$, as $t \to +\infty$.*

Condition (1) of the above theorem is also called *Lyapunov stability*, and the second condition is called *asymptotic stability*. Note that an equilibrium point (fixed point) is an invariant set.

In this section we shall assume that the function of our continuation problem (1) arises from a system of autonomous ordinary differential equations:

$$\dot{u} = f(u, \lambda), \qquad u \in \mathbb{R}^n, \quad \lambda \in \mathbb{R}, \tag{5}$$

where λ can be thought of as a parameter. The parameter is often regarded a fixed value, hence we often write (5) as

$$\dot{u} = f(u). \tag{6}$$

We shall see that the system (6) can be regarded as a dynamical system.

THEOREM 17. *Consider a system of ordinary differential equations*

$$\dot{u} = f(u), \qquad u \in \mathbb{R}^n,$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is smooth in an open region $U \subset \mathbb{R}^n$. Then there is a unique function $u = u(t, u_0)$, $u : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$, that is smooth in (t, u_0), and satisfies, for each $u_0 \in U$, the following conditions:

(1) $u(0, u_0) = u_0$;

(2) there is an interval $\mathcal{I} = (-\delta_1, \delta_2)$, *where* $\delta_{1,2} = \delta_{1,2}(u_0) > 0$, *such that, for all* $t \in \mathcal{I}$,

$$y(t) = u(t, u_0) \in U$$

and

$$\dot{y}(t) = f(y(t)).$$

The function $u(t, u_0)$ gives rise to two objects; a solution curve

$$\mathrm{Cr}(u_0) = \{(t, u) : u = u(t, u_0), t \in \mathcal{I}\} \subset \mathbb{R} \times \mathbb{R}^n$$

and an orbit

$$\mathrm{Or}(u_0) = \{u : u = u(t, u_0), t \in \mathcal{I}\} \subset \mathbb{R}^n.$$

Note that the orbit is the projection of the solution curve onto the state space $\mathbb{R}^n$. The evolution operator $\phi^t : \mathbb{R}^n \to \mathbb{R}^n$ can thus be written

$$\phi^t u_0 = u(t, u_0).$$

Thus we can regard the system of ordinary differential equations (6) as a dynamical system.

THEOREM 18. *Consider a dynamical system* (6), *where* f *is smooth. Suppose that it has an equilibrium* u_0, *then* u_0 *is stable if all eigenvalues* $\sigma_1, \sigma_2, \ldots, \sigma_n$ *of* $f_u(u_0)$ *satisfy* $\mathrm{Re}(\sigma) < 0$.

DEFINITION 19. *A dynamical system* $\{\mathcal{T}, \mathbb{R}^n, \phi^t\}$ *is called locally topologically equivalent to a dynamical system* $\{\mathcal{T}, \mathbb{R}^n, \psi^t\}$ *if there exists a homeomorphism* $h : \mathbb{R}^n$ *mapping orbits of the first system to orbits of the second system preserving the direction of time.*

DEFINITION 20. *A dynamical system* $\{\mathcal{T}, \mathbb{R}^n, \phi^t\}$ *is called locally topologically equivalent near an equilibrium* u_0 *to a dynamical system* $\{\mathcal{T}, \mathbb{R}^n, \psi^t\}$ *near an equilibrium* y_0 *if there exists a homeomorphism* $h : \mathbb{R}^n \to \mathbb{R}^n$ *that is*

(1) defined in a small neighborhood $U \in \mathbb{R}^n$ *of* u_0

(2) satisfies $y_0 = h(u_0)$

(3) maps orbits of the first system in U *onto orbits of the second system in* $V = h(U) \subset \mathbb{R}^n$, *preserving the direction of time.*

DEFINITION 21. *The appearance of a topologically nonequivalent phase portrait under variation of parameters is called a* bifurcation.

DEFINITION 22. *The* codimension *of a bifurcation system* (5) *is the difference between the dimension of the parameter space and the dimension of the corresponding bifurcation boundary.*

Bifurcations can be divided into two principal classes:

- Local bifurcations, which can be analyzed through changes in the local stability properties of equilibria, periodic orbits or other invariant sets as parameters cross through critical thresholds;
- global bifurcations, that cannot be detected purely by a stability analysis of the invariant set.

In this paper, we shall be concerned with local bifurcations only detecting local bifurcations along a solution branch of (5), and we shall use the term *bifurcation point* for points on the solution branch where the stability properties change.

3.1. Hyperbolic equilibria.

DEFINITION 23. *An equilibrium u_0 of* (6) *is called hyperbolic if there are no eigenvalues of $f_u(u_0)$ on the imaginary axis.*

For the system (5) we shall say that (u_0, λ_0) is hyperbolic if u_0 hyperbolic for $\lambda = \lambda_0$.

PROPOSITION 24. *Consider the system* (5) *where f and f_u are smooth near u_0. If (u_0, λ_0) is a hyperbolic equilibrium point of* (5) *then there exists a smooth curve $(u(s), \lambda(s))$ such that $f(u(s), \lambda(s)) = 0$ and $u(\lambda_0) = u_0$ for $|s|$ sufficiently small. Furthermore, for sufficiently small values of s, each point on the curve $(u(s), \lambda(s))$ is hyperbolic.*

PROOF. Note that a hyperbolic point is a regular point. The implicit function theorem therefore guarantees the existence of a smooth regular curve in some neighborhood of (u_0, λ_0). What is left to prove is that for sufficiently small values of s, the curve contains only hyperbolic points. Since u varies smoothly with respect to s, and f_u varies smoothly with respect to u, f_u varies smoothly with respect to s. The eigenvalues of $f_u(s)$ are determined by the characteristic equation $\det(f_u(s) - \sigma I) = 0$. This is is a polynomial in σ, where the coefficients are smoothly dependent of s. Since the roots of a polynomial depends continuously on the coefficients of the polynomial, the proposition is true. □

The above proposition states in other words that hyperbolic points persist for sufficiently small perturbations of the parameter λ.

3.2. Non-hyperbolic equilibria. The real part of an eigenvalue can reach zero in two ways; either $\sigma = 0$, or a complex conjugate pair of eigenvalues reach the imaginary axis, $\sigma = \pm a\boldsymbol{i}$. The first case corresponds to the appearance of a *fold bifurcation* and the second to a *Hopf bifurcation*. These are the two *generic* bifurcations we expect to meet at a one parameter analysis of (5). Another bifurcation that we shall consider occur at singular solution points. We shall limit our discussion to the simplest case where kernel of the Jacobian at the point is two dimensional.

Simple Folds. We shall restrict our discussion of folds to *simple folds*:

DEFINITION 25 (Simple fold). *A solution* (u_0, λ_0) *is called a* simple fold *if*

$$\dim \ker(f_u^0) = 1 \quad \textit{and} \quad f_\lambda^0 \notin \text{Range}\{f_u^0\}.$$

THEOREM 26. *Suppose* (u_0, λ_0) *is a simple fold. Then the equation*

$$f(u, \lambda) = 0,$$

has, for small s, *a unique solution branch*

$$(u(s), \lambda(s)) = (u_0 + s\phi + v(s), \lambda(s))$$

with

$$v(0) = 0, \qquad \lambda(0) = 0, \qquad \phi^* v(s) = 0.$$

Moreover, if f *is* k *times differentiable near* (u_0, λ_0) *then so is* $(u(s), \lambda(s))$.

PROOF. A proof of this can be found in [**3**]. □

DEFINITION 27. *A simple fold is called a simple quadratic fold if* $\lambda''(s_0) \neq 0$.

A simple quadratic fold is also often called a turning point. Note that a fold can be detected by monitoring the eigenvalues of the Jacobian.

Branch Points.

DEFINITION 28. *A point* x_0 *is called a* branch point *for the continuation problem* (1) *if* $f(x_0) = 0$ *and there are at least two different smooth curves satisfying* (1) *and passing through* x_0.

Note that branch points must happen at singular points. The simplest case of a singular point is the case of a *simple singular point*:

DEFINITION 29. *A solution* $x_0 \equiv x_(s_0)$ *along a solution branch* $x(s)$ *of* (1) *is called a* simple singular point *if* $f_x(x_0)$ *has rank* $n - 1$.

The question naturally arises if a simple singular point can be a branch point? A criteria will be derived here. Suppose that $x(s)$ is a smooth solution branch of $f(x) = 0$. Differentiation with respect to s yields

$$f_x(x(s))x'(s) = 0.$$

Assume that the direction vector $x'(s)$ is normalized, i.e $\| x'(s) \| = 1$. Let $x_0 \equiv x(s_0)$ be a simple singular point with ker $f_x^0 = \text{Span}\{\phi_1, \phi_2\}$ and $\ker((f_x^0)^*) = \text{Span}\{\psi\}$. The direction vector $x'(s)$ holds in particular at x_0, thus $x'(s)$ must be of the form

$$x'(s) = \alpha\phi_1 + \beta\phi_2, \tag{7}$$

for some constants α and β. Let us denote these by α_1 and β_1, respectively. If there is another branch passing through x_0 then there must be another pair of constants (α_2, β_2) such that (7) hold. By differentiating $f(x(s)) = 0$ twice, and evaluating at x_0 we obtain

$$f_x^0 x_0'' + f_{xx}^0 x_0' x_0' = 0.$$

Multiplying this by ψ^* we get

$$\psi^* f_{xx}^0 (\alpha\phi_1 + \beta\phi_2)(\alpha\phi_1 + \beta\phi_2) = 0,$$

which may be written as

(8) $$c_{11}\alpha^2 + 2c_{12}\alpha\beta + c_{22}\beta^2 = 0, \qquad c_{ij} = \phi^* f_{xx}^0 \phi_i \phi_j.$$

This equation (8) is called the *algebraic branching equation*. Suppose that $\alpha \neq 0$ then we can write (8) as

$$c_{11} + 2c_{12}\frac{\beta}{\alpha} + c_{22}(\frac{\beta}{\alpha})^2 = 0.$$

Hence, we see that if the discriminant $\Delta_0 \equiv c_{12}^2 - c_{11}c_{22}$ is positive then (8) has two distinct real nontrivial solution pairs (α_1, β_1) and (α_2, β_2) which are unique up to scaling. Note that the discriminant cannot be negative, since we already have one real root (α_1, β_1) corresponding to the direction vector x_0'. Thus if $\Delta_0 > 0$ we expect a second branch to pass through x_0.

DEFINITION 30. *A simple singular point x_0 having a positive discriminant Δ_0 is called* a simple branch point.

LEMMA 31. *Let $x(s)$ be a solution branch of* (1). *Consider the matrix*

$$D(s) = \begin{pmatrix} f_x(x(s)) \\ x'(s)^* \end{pmatrix}.$$

Then the scalar function

$$d(s) = \det D(s)$$

has a regular zero at $s = s_0$.

PROOF. A proof can be found in [**9**]. □

Note that this gives us means for detecting branch points.

Hopf Bifurcations.

THEOREM 32 (The Hopf Bifurcation Theorem). *Suppose that along a stationary solution branch $(u(\lambda), \lambda)$, of* (5) *a complex conjugate pair of eigenvalues*

$$\alpha(\lambda) \pm \mathbb{i}\beta(\lambda),$$

of $f_u(u(\lambda), \lambda)$ crosses the imaginary axis transversally, i.e., for some λ_0,

$$\alpha(\lambda_0) = 0, \quad \beta(\lambda_0) \neq 0, \quad \textit{and} \quad \frac{\mathrm{d}\alpha(\lambda_0)}{\mathrm{d}\lambda} \neq 0.$$

Also assume that there are no other eigenvalues on the imaginary axis.

Then there is a Hopf bifurcation*, i.e., a family of periodic solutions bifurcate from the stationary solution at* (u_0, λ_0).

PROOF. The theorem is stated in [**3**]. No references to the proof is given there. □

A Hopf bifurcation point can therefore be detected by monitoring the eigenvalues.

3.3. Periodic Solutions. A solution $v(t)$ of (5) is called periodic if v also satisfies

$$v(0) = v(T),$$

for some *period* $T \in \mathbb{R}$. Note that a periodic solution is a cycle. Suppose that $v(t)$ denotes a periodic solution of (5) of period T, and let $\delta(t)$ denote a perturbation from the periodic solution. We shall consider the curve $w(t) = v(t) + \delta(t)$. By linearizing (5) about $v(t)$ we obtain

$$\begin{aligned} \dot{w}(t) &= f(v(t) + \delta(t)) = f(v(t)) + f_u(v(t)(w(t) - v(t)) + \mathcal{O}(\| w(t) - v(t) \|^2) \\ &= A(t)u(t) + \mathcal{O}(\| u(t) \|^2). \end{aligned} \tag{9}$$

Truncating $\mathcal{O}(\| u(t) \|^2$ terms results in the linear T-periodic system

$$\dot{v} = A(t)v, \qquad v \in \mathbb{R}^n,$$

where $A(t) = f_u(v(t)), A(t + T) = A(t)$. The system (3.3) is called the *variational equation* of the periodic solution $v(t)$. It is clear that the stability of $v(t)$ depends on the properties of the variational equation.

DEFINITION 33. *The time-dependent matrix $M(t)$ is called the* fundamental matrix solution *of* (5) *if it satisfies*

$$\dot{M} = A(t)M,$$

with the initial condition $M(0) = I_n$.

Note that any solution $v(t)$ of (3.3) satisfies

$$v(T) = M(T)v(0).$$

The matrix $M(T)$ is called the *monodromy matrix* of the periodic solution $v(t)$, and its eigenvalues are called the *Floquet multipliers*. Note that 1 is always a Floquet multiplier since $M(T)u(0) = u(0)$. Bifurcation points of branches of periodic solution can be studied by monitoring the Floquet multipliers along the solution branch. We will not go into the types of bifurcations that arise in the context of periodic solutions, as it is not part of the scope of this paper. Further details on the topic can be found in [**9**].

CHAPTER 3

Predictor-Corrector Methods

There are at least two different approaches for obtaining the solution curve $x(s)$, namely the techniques of *piecewise linear continuation* and *predictor-corrector methods*. Only predictor-corrector (PC) methods will be discussed in this paper.

A predictor-corrector method generally consists of the following steps

(1) prediction of the next point,

(2) correction,

(3) step-size control,

which will be described in the below sections.

1. Prediction

Suppose that a solution point x_0,the direction vector v_0 and some initial step-length h_0 is known, then the next point, $\tilde{x}$ point may be guessed by the *Euler prediction*.

$$\tilde{x} = x_0 + h_0 v_0.$$

Let $A = f_x(x_0)$. To obtain an initial direction vector note that A^* can be factored into

$$A^* = Q \begin{pmatrix} R \\ 0 \end{pmatrix},$$

hence if z denotes the last column of Q, then $Az = 0$, and since $Q^* Q = I$ and $\| z \| = 1$, z can be used as an initial direction vector. In fact, if we wish that z should be the tangent vector it suffices to note that

$$(A^*, z) = Q \begin{pmatrix} R & 0 \\ 0^* & 1 \end{pmatrix},$$

hence

$$\det \begin{pmatrix} A \\ z^* \end{pmatrix} = \det(A^*, z) = \det Q \det R.$$

Subsequent direction vectors can be obtained by solving

$$\begin{pmatrix} f_x^1 \\ v_0* \end{pmatrix} \begin{pmatrix} v_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

or by normalizing

$$v_i = \frac{x_i - x_{i-1}}{h_{i-1}}$$

Also note that higher order predictors can also be used. See [**6**] for a discussion of higher order predictors based on interpolation polynomials and corresponding steplength strategies.

2. Correction

The purpose of the correction step is to iteratively obtain better and better approximation of the next point until some predefined tolerances are met. Generally it is advantageous to utilize Newton (or Newton-like) methods to obtain better approximations. Two methods are considered in this paper, namely the *pseudo-archlength continuation* and the *Gauss-Newton method.*

2.1. Newton's method.

Let $F : \mathbb{R}^k \to \mathbb{R}^k$, $x \mapsto y$, be a smooth function, and let $\tilde{x}$ be an approximation of a root of F. Linearizing F about $\tilde{x}$ we obtain

$$F(\tilde{x}) + F_x(\tilde{x})(x - \tilde{x}) = 0. \tag{10}$$

If the matrix F_x is invertible the linear system will have the solution

$$x = \tilde{x} - F_x^{-1}(\tilde{x})F(\tilde{x}). \tag{11}$$

Note that we can define the following Newton iterations

$$\mathcal{N}(\tilde{x}) = \tilde{x} + \Delta x, \tag{12}$$

where the displacement vector Δx is obtained by solving

$$F_x(\tilde{x})\Delta x = -F(\tilde{x}).$$

THEOREM 34. *Suppose the system $F : \mathbb{R}^k \to \mathbb{R}^k$, $x \mapsto y$, is smooth and has an equilibrium $x = \hat{x}$ with no zero eigenvalue of the Jacobian matrix $F_x(\hat{x})$. Then there is a neighborhood U of $\hat{x}$ such that the Newton iterations* (12) *converge to $\hat{x}$ from any initial point $\tilde{x} \in U$. Moreover*

$$\| \mathcal{N}^{j+1}(\tilde{x}) - \hat{x} \| \leq \kappa \| \mathcal{N}^j(\tilde{x}) - \hat{x} \|^2, \qquad j = 0, 1, 2, \ldots,$$

for some $\kappa > 0$, uniformly for $\hat{x} \in U$.

2.2. Pseudo-arclength continuation. The pseudo-arclength approach can be considered a part of a more general class of methods where the basic approach is to append an equation $g(x) = 0$ to the system $f(x) = 0$ such that the Jacobian of the extended system (13) has full rank.

$$F(x) = \begin{pmatrix} f(x) \\ g(x) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{13}$$

Geometrically, the idea of the pseudo-arclength continuation, is that the predicted point is projected orthogonally to the unit direction vector v_0 of the previous solution point x_0 onto the solution curve. Mathematically this may be stated by

$$g(x) = \langle x - \tilde{x}, v_0 \rangle = (x - \tilde{x})^* v_0. \tag{14}$$

Often the predictor step is incorporated into the pseudo-arclength equation ,

$$g(x) = \langle x - x_0 + h v_0, v_0 \rangle.$$

If $\| v_0 \| = 1$, then we obtain

$$g(x) = \langle x - x_0, v_0 \rangle - h$$

or

$$\begin{aligned} g(x) &= \langle u - u_0, \dot{u}_0 \rangle + (\lambda - \lambda_0)\dot{\lambda}_0 - h \\ &= (u - u_0)^* \dot{u}_0 + (\lambda - \lambda_0)\dot{\lambda}_0 - h. \end{aligned}$$

THEOREM 35. *The Jacobian of the pseudo-arclength system* (13) *is nonsingular at a regular solution point.*

PROOF. A proof can be found in [**2**]. □

2.3. Gauss-Newton method. The Gauss-Newton method is another projection of the predictor $\tilde{x}$ onto the solution branch $x(s) \in f^{-1}(0)$. More specifically we seek to find the point on $\hat{x} \in x(s)$ such that

$$\hat{x} = \arg\min_{w} \{ \| \tilde{x} - w \| \; | f(w) = 0 \}. \tag{15}$$

Using the theory of Lagrange multipliers we get that $\hat{x}$ must satisfy

$$\begin{gathered} f(\hat{x}) = 0 \\ \hat{x} - \tilde{x} = f_x(\hat{x})^* \lambda, \end{gathered}$$

where λ is the Lagrange multiplier. The second condition can be seen to be equivalent to

$$t(f_x(\hat{x}))^* (\hat{x} - \tilde{x}) = 0.$$

Linearizing about $\tilde{x}$, and ignoring higher order terms, we obtain:

$$\begin{gathered} f(\hat{x}) \approx f(\tilde{x}) + f_x(\tilde{x})(\hat{x} - \tilde{x}) \\ t(f_x(\hat{x}))^* (\hat{x} - \tilde{x}) \approx t(f_x(\tilde{x})^* (\hat{x} - \tilde{x}) \end{gathered}$$

Hence we are left with solving the system

$$f(\tilde{x}) + f_x(\tilde{x})(\hat{x} - \tilde{x}) = 0 \tag{16}$$

$$t(f_x(\tilde{x})^*(\hat{x} - \tilde{x}) = 0. \tag{17}$$

We shall see that this system can be solved in a Newton like manner, in terms of using the *Moore-Penrose* inverse.

DEFINITION 36. *Let A be an $n \times (n+1)$ matrix with maximal rank. Then the* Moore-Penrose *inverse of A is defined by $A^+ = A^*(AA^*)^{-1}$.*

LEMMA 37. *Let A be an $n \times (n+1)$ matrix with maximal rank, and let $t(A)$ be its tangent vector. Then the following statements are equivalent for all $b \in \mathbb{R}^n$ and $x \in R^{n+1}$:*

(1) $Ax = b$ and $t(A)^*x = 0$;

(2) $x = A^+b$;

PROOF. See [**6**] for a proof. □

Hence if $\tilde{x}$ is a regular point of f. Then we can obtain the next Gauss-Newton point $\mathcal{G}(\tilde{x})$ by

$$\mathcal{G}(\tilde{x}) = \tilde{x} - f_x(\tilde{x})^+ f(\tilde{x}). \tag{18}$$

THEOREM 38. *Let $f : \mathbb{R}^{n+1} \to \mathbb{R}^n$ be a smooth map having zero as a regular value. Then there exists an open neighborhood $U \supset f^{-1}(0)$ such that the following assertions hold.*

(1) The solution map $\tilde{x} \mapsto \mathcal{S}\tilde{x} \in f^{-1}(0)$ such that $\mathcal{S}(\tilde{x})$ solves the minimization problem (15) *is uniquely defined and smooth.*

(2) For each $x \in U$ the Gauss-Newton sequence $\{\mathcal{G}^i(\tilde{x})\}_i^\infty$ converges to a point in $f^{-1}(0)$.

(3) The following estimates hold locally uniformly for $x \in U$

(a) $\| \mathcal{G}^2(\tilde{x}) - \mathcal{G}(\tilde{x}) \| = \mathcal{O}(\| \mathcal{G}(\tilde{x}) - \tilde{x} \|^2)$;

(b) $\| \mathcal{G}^\infty(\tilde{x}) - \mathcal{G}(\tilde{x}) \| = \mathcal{O}(\| \mathcal{G}^\infty(\tilde{x}) - \tilde{x} \|^2)$;

(c) $\| \mathcal{G}(\tilde{x}) - \mathcal{S}(\tilde{x}) \| = \mathcal{O}(\| \tilde{x} - \mathcal{S}(\tilde{x}) \|^2)$;

(d) $\| \mathcal{G}^\infty(\tilde{x}) - \mathcal{S}(\tilde{x}) \| = \mathcal{O}(\| \tilde{x} - \mathcal{S}(\tilde{x}) \|^2)$.

(4) The relation $\mathcal{G}(U) \subset U$ holds.

PROOF. Proof of this can be found in [**6**]. □

It is numerically inefficient to obtain the matrix $f_x(\tilde{x})^+$ at each iteration step. Allgower et.al [**6**] describes a method for the calculation of the gauss-newton step based on QR or LU factorizations of the Jacobian, such that the Moore-Penrose inverse do not explicitly need to be calculated. The approach taken in CONTBIF is to use Newton's method directly on the equations (16) and (17) instead. The advantage of doing this is that we can employ sparse solvers to the system (16) and (17), which we expect to be more effective with regard to the computation time.

2.4. Convergence criteria. There are several possibilities for acceptance criteria for Newton's or Gauss-Newton's method. We shall list some possibilities here

$$\| \Delta x \| < \epsilon_x, \tag{19}$$

$$\| f(x_1) - f(x_0) \| < \epsilon_f, \tag{20}$$

$$\| \Delta x \|_\infty < \epsilon_{x,\infty} \tag{21}$$

$$\| f(x_1) - f(x_0) \|_\infty < \epsilon_{f,\infty} \tag{22}$$

$$\frac{|\Delta\lambda|}{1 + |\lambda|} < \epsilon_\lambda \tag{23}$$

$$\frac{\| \Delta u \|_\infty}{1 + \| u \|_\infty} < \epsilon_u. \tag{24}$$

The reader should observe that (19) and (20) are more sensitive to the dimension n of the problem. In CONTBIF all of the above criteria are implemented, and the user may select which should be active for a particular continuation.

3. Stepsize control

There are many ways to introduce steplength adaption. These are mainly based on the performance of the Newton iterations. A simple scheme may be that we increase the steplength whenever few Newton iterations were needed to compute the last point, and conversely we decrease the steplength when the previous point needed many Newton iterations. In CONTBIF there is a limit to the number of Newton iterations that are allowed in the corrector process. Roughly the steplength is increased if the number of iterations were bigger than $2/3$'s of the limit, and decreased if the number of iterations were less that a factor of $1/3$ of the limit. Otherwise the previous steplength is kept. The increase and decrease factors may be chosen by the user. Other methods such as *asymptotic expansion* and *Den Heijer & Reinbolt steplength adaption* are discussed for Newton-Gauss iterations in [**6**].

4. Computing the initial equilibrium point

If an initial root (u_0, λ_0) of f, is not known we need a way to approximate it. This may be done by Newton iterations on the system $f(u) = 0$. This however requires that our guess is within the region of attraction for the Newton method. A second approach is to define a smooth *homotopy* function $H : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ such that $H(u, 1) = G(u)$ and $H(u, 0) = f(u)$ where $G(u)$ is a trivial smooth map having known zero points [**6**], and attempt to trace out the implicitly defined curve $c(s) \in H^{-1}(0)$ from a starting point $(u_1, 1)$ to a solution point $(u_0, 0)$. As an example we may consider a *convex homotopy* function such as

$$H(u, \lambda) = \lambda G(u) + (1 - \lambda) f(u).$$

5. Computing special points on the curve

Suppose that we have a smooth scalar *test function* $\phi(x)$, and we wish to compute point $x \in f^{-1}(0)$ such that $\phi(x) = 0$. Suppose that x_i and x_{i+1} are two consecutive point on the curve $x(s) \in f^{-1}(0)$, and let s_i, s_{i+1} be such that $x(s_i) = x_i$ and $x(s_{i+1}) = x_{i+1}$. If

$$\phi(x_i)\phi(x_{i+1}) < 0,$$

a zero of ϕ has been detected for some point on the curve $x(s) \in f^{-1}(0)$ for some value s such that $s_i < s < s_{i+1}$. In order to approximate the special point the following system may be solved

$$\begin{aligned} f(x) &= 0 \\ \phi(x) &= 0 \end{aligned}$$

using Newton iterations. The drawback of using this method is that we need to calculate derivatives of $\phi(x)$ which is not always easy to do. Instead the special point can be solved by using a one-dimensional secant method in combination with newton correction ensuring that $f(x) = 0$ at each point. Note that the one dimensional Newton method can be written as

$$\phi'(x(s_{i+1}))\Delta s = -\phi(x(s_{i+1})). \tag{25}$$

The secant method is essentially a discrete version of (25):

$$\Delta s = -\frac{s_{i+1} - s_i}{\phi(x(s_{i+1})) - \phi(x(s_i))}\phi(x(s_{i+1})).$$

The predictor for the next point thus becomes $\tilde{x}_{i+2} = x_{i+1} + \Delta s x'(s)$, and we must perform successive correction steps to find x_{i+2}.

CHAPTER 4

Deriving the basic continuation equations

The focus of CONTBIFis to be able to perform a one parameter bifurcation analysis of autonomous dynamical systems, in explicit equations of the form

$$\dot{u} = f(u, \lambda) \tag{26}$$

where $\lambda \in \mathbb{R}$ is a parameter.

1. Continuation of equilibria

Setting up the system for continuation of equilibria is trivial, the problem to be regarded is simply

$$f(u, \lambda) = 0. \tag{27}$$

1.1. The initial point.

Generally CONTBIFassumes that an initial equilibrium point is known for a fixed parameter value to within the radius of Newton convergence. If this is not the case, the user could define a *homotopy* function and using numerical continuation to obtain an initial point.

1.2. Detection of special points.

Generally we are interested in detecting the following special points; simple fold, Hopf bifurcation point and a simple branch point. To detect a fold it is enough to keep an eye on the number of eigenvalues in $\mathbb{C}^+$ and $\mathbb{C}^-$, however to be able to take advantage of the secant approach the following function may be considered

$$\zeta_F^1(u, \lambda) = \prod_i \sigma_i(u, \lambda), \tag{28}$$

where σ_i are the eigenvalues of $f_u(u, \lambda)$. Noting that the function (28) is simply the determinant of $f_u(u, \lambda)$, we may suggest the test function

$$\zeta_F^2 = \det f_u(u, \lambda). \tag{29}$$

If are interested in simple quadratic folds the following test function may also be considered

$$\zeta_F^3(\lambda) = \lambda'(s). \tag{30}$$

Another approach that we will not discuss here is to monitor for extremal values of λ.

Detection of Hopf bifurcations may be done by considering the real parts of the eigenvalues by defining the test function:

$$\zeta_H^1 = \prod_{i>j}(\sigma_i(u,\lambda) + \gamma_j(u,\lambda)) \tag{31}$$

A second approach that does not require evaluation of eigenvalues is suggested in [**9**] and implemented in CL_MATCONT.

Recalling the lemma 31, we see that a test function for a simple branch point is

$$\zeta_B = \det\begin{pmatrix} f_x(x) \\ x'(s) \end{pmatrix}. \tag{32}$$

Another suggestion is to use the above described test functions for detecting the bifurcation points, and then to use the function

$$\zeta = \min_i |\mathrm{Re}\{\sigma_i\}|,$$

to locate the bifurcation points more precisely. Note that this function is not smooth, however we may expect it to be smooth in a sufficiently small neighborhood of the bifurcation point. This function has been tested to work for the low dimensional examples in this paper. Also note that this test function converges to any bifurcation point, hence if two of more bifurcation points are close (with respect to the steplength) this test function may fail to converge. Also note that it is not necessary to evaluate the test functions at each point of the continuation process. Generally, a bifurcation point may be detected by counting the number of stable eigenvalues at each point. A change in this number, indicates that a bifurcation point has been passed, and we may use the test functions above to determine the type of bifurcation point to look for.

Also note that the function (32) is sufficient for detecting the existence of a simple branch point, but difficulties may arise if we use the secant approach to locate it more accurately. This is because both the pseudo-arclength system (13) and the Gauss-Newton system (16) and (17) are singular at the branch point. In fact it can be shown that the domain of convergence for the Newton (or Gauss-Newton) corrections shrinks as one approaches the branching point, see [**6**]. To overcome this obstacle, we need to extend the system such that the branch point becomes a regular point of the extended system. Moore [**11**] suggested using Newton iterations on the following system

$$\begin{aligned} f(u,\lambda) + \mu\xi &= 0; \\ f_u(u,\lambda)^*\xi &= 0; \\ \xi^* f_\lambda(u,\lambda) &= 0; \\ \xi^*\xi - 1 &= 0. \end{aligned}$$

Initial estimates for this system is obtained by letting $\mu = 0$, and setting ξ to be the eigenvector of $f_u(\tilde{x})$ that corresponds the eigenvalue of the smallest absolute value. Here $\tilde{x}$ is simply some point close to the branch point.

We must also address the task of switching branch at a simple branch point. Recall that the kernel of f_x at a simple branch point x_0 is two dimensional. The basis vectors of the kernel can therefore be chosen such that $\phi_1 = t(f_x^0)$, and $\phi_2 \perp \phi_1$. Then $(\alpha_1, \beta_1) = (1, 0)$ is a root of (8) and $c_{11} = 0$. Under the assumption that $\Delta_0 > 0$ we also have that $c_{12} \neq 0$. Thus we find that the second root satisfies

$$\frac{\alpha_2}{\beta_2} = -\frac{c_{22}}{2c_{12}}. \tag{33}$$

To evaluate c_{12} and c_{22} we need to find ϕ_2 and ψ. Since we have defined $\phi_2 \perp \phi_1$, ϕ_2 is a null vector of

$$\begin{pmatrix} f_x^0 \\ t(f_x^0)^* \end{pmatrix}. \tag{34}$$

Similarly it can be seen that ψ can be computed by

$$\begin{pmatrix} f_x^0 \\ t(f_x^0)^* \end{pmatrix}^* \begin{pmatrix} \psi \\ 0 \end{pmatrix} = 0 \tag{35}$$

We are free to choose one value of a_2 and hence β_2 can be determined. The resulting new direction vector thus becomes

$$x_0' = \alpha_2\phi_1 + \beta_2\phi_2, \tag{36}$$

and as always we must remember to normalize it. Generally we may prefer to avoid the calculation of f_{xx}^0, in such cases the *orthogonal direction method* can be used. The idea behind this method is to use ϕ_2 as an approximation of the tangent vector x_0'. The point of convergence for the branch detection algorithm is only an approximation of a branch point, hence the matrix (34) is generally non-singular. In practice the system (35) must therefore be replaced by a minimization problem

$$\min_{\phi_2}\{\| f_x^0\phi_2 \|^2 + (\phi_1^*\phi_2)^2 | \phi_2^*\phi_2 = 1\}.$$

Let A denote the matrix $\begin{pmatrix} f_x^0 \\ t(f_x^0) \end{pmatrix}$. Then a solution to the above minimization problem is the unit eigenvector of A^*A corresponding to the smallest eigenvalue [**6**].

2. Continuation of Periodic Solutions

In this section we shall assume that an initial periodic solution, $v(t)$, of (5) is known, and we discuss how to obtain the next periodic solution $w(t)$. Certainly, a periodic solution must satisfy

$$\dot{w}(t) = f(w(t), \lambda) \tag{37}$$

and

$$w(0) = w(T), \tag{38}$$

where T is the period of the periodic solution. The period is fixed by rescaling

$$t \mapsto \frac{t}{T} \tag{39}$$

hence the equations (37) and (38) becomes

$$w(t) = Tf(w(t), \lambda), \tag{40}$$

$$w(0) = w(1). \tag{41}$$

The above equations do not uniquely determine w and T, since if $w(t)$ is a periodic solution, then $w(t + \delta)$ is also a periodic solution for any real number δ. Thus a *phase condition* is needed. The phase can be 'anchored' by the *Poincarè orthogonality condition*

$$(w(0) - v(0))^* \dot{v}(0) = 0. \tag{42}$$

By plotting the curves $v(t)$ and $w(t)$ in the same diagram, it can be seen that this condition (42) allows sharp peaks in the solution to move in the continuation [**4**]. Although theoretically unimportant, numerically this is unfortunate the the mesh must be re-adapted more often in order to obtain the best accuracy. To remedy this, one may consider obtaining the solution $\tilde{w}(t + \hat{\sigma})$ such that

$$\hat{\sigma} = \arg\min_{\sigma}\{g(\sigma)\},$$

where

$$g(\sigma) = \int_0^1 \parallel \tilde{w}(t + \sigma) - v(t) \parallel_2^2 \mathrm{d}t.$$

Which by differentiation and integration by parts becomes

$$\int_0^1 w(t)^* \dot{v}(t) \mathrm{d}t = 0, \tag{43}$$

where we have made the substitution $w(t) = \tilde{w}(t + \hat{\sigma})$.

Together the equations (40), (41) and (43) form a boundary value problem. There are several techniques that can be used for solving this system, including those of *single shooting*, *multiple shooting* and *finite differences*. However the most reliable choice has proven to be *orthogonal collocation* [**9**].

2.1. Orthogonal Collocation. Introduce a mesh

$$\{0 = t_1 < t_2 < \cdots < t_N + 1 = 1\},$$

where

$$h_j \equiv t_{j+1} - t_j, \qquad (1 \leq j \leq N),$$

and define the space of vector piecewise polynomials $\mathcal{P}_h^m$ as

$$\mathcal{P}_h^m = \{p_h \in C[0, 1] : p_h|_{[}t_{j-1}, t_j] \in \mathcal{P}^m\},$$

where $\mathcal{P}^m$ is the space of vector polynomials of degree $< m$.

REMARK 39. *Note that $\mathcal{P}^m$ is a vector space.*

The first idea is to limit the space of solution functions $w(\cdot)$ to that of $\mathcal{P}_h^m$. We need to locally construct a polynomial of maximal degree $\leq m$ in each interval $[t_{-1}, t_j]$. To uniquely define such a polynomial we need to know $m+1$ points in this interval, which we shall denote by $t_j = t_{j,1} < \cdots < t_{j,i} < \cdots < t_{j,m+1} = t_{j+1}$ For simplicity we shall assume they are distributed uniformly, i.e

$$t_{j,i} \equiv t_j + \frac{i}{m} h_j.$$

With this settled we can introduce the Lagrange basis polynomials, $\{l_{j,i} : j = 1, \cdots, N \quad i = 1, 2, \cdots, m+1\}$, on each subinterval $[t_j, t_{j+1}]$, defined by

$$l_{j,i} = \prod_{k=1, k\neq i}^{m+1} \frac{t - t_{j,k}}{t_{j,i} - t_{j,k}}.$$

The local polynomials may thus be written

$$p_j(t) = \sum_{i=1}^{m+1} l_{j,i} w_{j,i},$$

where $w_{j,i} \equiv w(t_{j,i})$.

The next step is to select m collocation points, $z_{j,i}$ in each interval, such that $t_j < z_{j,1} < \cdots < z_{j,m} < t_{j-1}$. In particular, the collocation points must be chosen to be *Gauss* points in the particular interval to obtain the best accuracy of the method [**9**]. Our problem is then reduced to finding the piecewise polynomial $p = \cup_j p_j$ such that

$$p_j(z_{j,i})' - Tf(p_j(z_{j,i}), \lambda) = 0,$$

for $j = 1, \ldots, N$, and $i = 1, \ldots, m$, and such that p satisfies the boundary condition (38) and the integral constraint (43).

The problem is well defined [**2**], and the order of accuracy of the orthogonal collocation method is $\parallel p_h - w \parallel_\infty = \mathcal{O}(h^m)$, while at the meshpoints t_j there is super-convergence: $\max_j |p_h(t_j) - w(t_j)| = \mathcal{O}(h^{2m})$.

To obtain a discrete version of the phase condition (43) a natural choice is to used closed Newton-Cotes quadrature formulas, in explicit we can write

$$\begin{aligned} \int_0^1 w(t)\dot{v}(t)\mathrm{d}t &= \sum_{j=1}^{N} \int_{t_{j,1}}^{t_{j,m+1}} w(t)\dot{v}(t)\mathrm{d}t \\ &\approx \sum_{j=1}^{N} \sum_{i=1}^{m+1} \omega_{j,i} w_{j,i} \dot{v}_{j,i} = 0, \end{aligned} \tag{44}$$

where $\omega_{j,i}$, $i = 1 \ldots m+1$, $j = 1 \ldots N$ are the quadrature weights.

To summarize the discrete system to be analyzed takes the form

$$p_j'(z_{j,i}) - Tf(p_j(z_{j,i}, \lambda)) = 0, \tag{45}$$

$$w_{1,1} - w_{N,m+1} = 0, \tag{46}$$

$$\sum_{j=1}^{N} \sum_{i=1}^{m+1} \omega_{j,i} w(t_{j,i})^* \dot{v}(t_{j,i}) = 0. \tag{47}$$

2.2. Obtaining an initial limit cycle from a Hopf bifurcation point. Near a Hopf bifuration (u_0, λ_0) of (5), the following asymptotic estimates hold [**3**]:

$$v(t;\epsilon) = u_0 + \epsilon\phi(t) + \mathcal{O}(\epsilon^2), \tag{48}$$

$$T(\epsilon) = T_0 + \mathcal{O}(\epsilon^2), \tag{49}$$

$$\lambda(\epsilon) = \lambda_0 + \mathcal{O}(\epsilon^2), \tag{50}$$

where ϵ locally parameterizes the branch of periodic solutions. $T(\epsilon)$ denotes the period, and

$$T_0 = \frac{2\pi}{\psi_0},$$

where $\pm\mathbb{i}\psi_0$ are the only eigenvalues on the imaginary axis. The function ϕ is the normalized nonzero periodic solution of the linearized constant coefficient problem

$$\begin{aligned} \phi'(t) &= f_u(u_0, \lambda_0)\phi(t), \\ \phi(0) &= \phi(T). \end{aligned}$$

The above equation must be replaced by its time-scaled variant

$$\phi'(t) = Tf_u(u_0, \lambda_0)\phi(t), \tag{51}$$

$$\phi(0) = \phi(1). \tag{52}$$

Let $\eta_c \pm \mathbb{i}\eta_s$ denote the complex eigenvectors associated with the eigenvalues $\pm\mathbb{i}\psi_0$, then it is straightforward to check that a solution of (51) is

$$\phi(t) = \cos(2\pi t)\eta_c - \sin(2\pi t)\eta_s. \tag{53}$$

By the approximations (48),(49), and (50) the choice for an initial periodic solution v is given by

$$\begin{pmatrix} v(t) \\ T_v \\ \lambda_v \end{pmatrix} = \begin{pmatrix} u_0 + \epsilon\phi(t) \\ T_0 \\ \lambda_0 \end{pmatrix}.$$

Similarly the tangent vector $t(v) = v'(\epsilon)$ is approximated by

$$t(v(t)) = \begin{pmatrix} \phi(t) \\ 0 \\ 0 \end{pmatrix}.$$

In practice, it is advised to choose a small amplitude ϵ for the inital periodic solution. The first accepted periodic solution will be determined by a predictor-corrector step to the initial periodic solution. Also note that the phase condition for the zeroth step is modified to

$$\sum_{j=1}^{N}\sum_{i=1}^{m+1} \omega_{j,i} w(t_{j,i})^* \dot{\phi}(t_{j,i}) = 0. \tag{54}$$

3. Mesh adaptation

A look ahead at the examples 4 and 5 in the next chapter reveals the need for mesh adaptation. Generally, once would suspect that a mesh that is uniformly distributed with respect to the arclength would perform much better than a fixed mesh in time. Some attempts have been made to utilize the theory of moving mesh techniques to adapt the mesh. A few considerations will be made here.

We shall briefly review some moving mesh equations derived from the equidistribution principle. The general idea is to *equidistribute* some measure of the *discretization error* uniformly within each element. The measure of the error in the element (t_i, t_{i+1})can be formulated as

$$\int_{t_i}^{t_{i+1}} M(t,s)\mathrm{d}t, \tag{55}$$

where M is some *monitor function*. For the present discussion we shall use the following monitor function

$$M(t,s) = \| \dot{u}(t,s) \|, \tag{56}$$

where s can be taken to be the arclength parameter of the solution branch of the discrete system (45). The equidistribution principle can then be formulated as

$$\int_{t_i}^{t_{i+1}} M(t,s)\mathrm{d}t = \frac{1}{N}\int_0^1 M(t,s)\mathrm{d}t. \tag{57}$$

In order to formulate equations for continuous mesh adaption, we introduce a *computational domain* $\xi \in [0,1]$, such that the mesh is uniformly distributed on the computational domain, and introduce the *mesh function* $t(\xi,s)$. Note that the boundary conditions

$$t(0,\lambda) = 0; \tag{58}$$

and

$$t(1,\lambda) = 1, \tag{59}$$

must also be satisfied. We must also choose the initial mesh $t(\xi,0)$. In the implementation the initial mesh is chosen to be uniformly distributed, i.e $t(\xi,0) \equiv 1$. The equidistribution principle can now be reformulated as

$$\int_0^{t(\xi,s)} M(t,s)\mathrm{d}t = \xi\theta(s), \tag{60}$$

where

$$\theta(s) = \int_0^1 M(t,s)\mathrm{d}t. \tag{61}$$

From this principle, discrete versions of moving mesh equations can be deduced [**8**]. We shall consider the following

$$-\frac{E_i}{\tau} = \frac{1}{2h^2}\Big[(M_{i+1}+M_i)(t'_{i+1}-t'_i) - (M_i+M_{i-1})(t'_i - t'_{i-1})\Big], \tag{62}$$

$$-\frac{E_i}{\tau} = -t'_i, \tag{63}$$

$$-\frac{E_i}{\tau} = \frac{t'_{i+1}-2t'_i+t'_{i-1}}{h^2}, \tag{64}$$

$$\begin{aligned} -\frac{E_i}{\tau} = \frac{1}{2h^2}\Big[(M_{i+1}+M_i)(t'_{i+1}-t'_i) - (M_i+M_{i-1})(t'_i - t'_{i-1})\Big] \\ -2E_i\frac{t'_{i+1}-t'_{i-1}}{t_{i+1}-t_{i-1}}, \end{aligned} \tag{65}$$

where

$$E_i = \frac{1}{2h^2}\Big[(M_{i+1}+M_i)(t_{i+1}-t_i) - (M_i+M_{i-1})(t_i - t_{i-1})\Big]. \tag{66}$$

Note that $t' \equiv t'(s)$ in the above equations. Also note that τ is a relaxation constant with respect to s, due to a time scaling $s \mapsto s+\tau$, $0 < \tau << 1$. The key impact of introducing the relaxation parameter is that the mesh will be driven towards equidistribution even though the monitor function, and hence u does not vary with s. Also note that h is the length of the interval of an element in the computational domain, in explicit $h = \xi_{i+1} - \xi_i$ for any $i = 1 \dots N-1$. The above equations (62) to (65) can generally be expressed in the form

$$B(t,s)t' = -\frac{E(t,s)}{\tau}. \tag{67}$$

The matrix B is invertible and hence we can write

$$t' = \frac{1}{\tau}B(t,s)^{-1}E(t,s). \tag{68}$$

The first thought was to append the equation (68) to the system (45) to incorporate the moving mesh equation into the system. More precisely this amounted to recalculating the internal nodes and collocation points every time the system was evaluated, as well as to interpolate u. Suppose that an initial periodic solution has been computed, then we can evaluate the interpolation approximation of u with respect to t. The challenge is that we do not generally know u as a function of s. However since the equation (67) does not strictly depend on s the hope was that the mesh would be driven towards equidistribution anyway. This approach was tested on the Brusselator (example 5), but without satisfactory results. A second approach was therefore considered instead; namely to use the equation (68) to adapt the mesh at each step after a new periodic solution had been computed, and then to interpolate the already obtained solution onto the new mesh before continuing the continuation process. Note that the mesh is assumed to be equistributed if the speed $t'(s) = 0$. Using this we see that (67) reduces to the requirement that

$$E(t,s) = 0. \tag{69}$$

The idea was then to use Newton's method on this equation (69) to obtain a good mesh for the current periodic solution before advancing to the next solution point. This method worked a little better than the moving mesh approach above, but still in the case of the example 5 it performed worse than the fixed mesh. The reason for this is related to the steplength for the continuation process, if the steplength is too big we risk that the mesh used on the new solution is outside the region of convergence for Newtons method. We have tested this technique on the Brusselator in example 5 in chapter 7.

REMARK 40. *Note that the first approach is not included in the current version of CONTBIF, but the second is.*

CHAPTER 5

Special topics regarding the implementation

In this chapter we shall discuss the choices that are made for the implementation of CONTBIF as well as some aspects conserning optimization of the code.

1. Strategy for direction vector

There have been presented two strategies for the direction vector. Either the next direction vector can be aligned with the previous one, which seems to be the choice in programs such as AUTO and MATCONT, or we may calculate the tangent direction at each point and use $dt(f_x)$, where $d = \pm 1$ is a scalar deciding the direction we want to traverse the solution branch. This implementation falls in line with the first approach uses the first approach. The main reason for doing this is with regard to the computation of limit cycles, where there are no detection of bifurcation points are implemented. However, the reader should note that using the tangent direction simplifies the detection of branch points as the following test function is applicable:

$$t(f_{x_1})^* t(f_{x_0}) < 0. \tag{70}$$

This follows directly from lemma 31 and the definition of the tangent vector 6, but it also requires that the tangent direction is obtained at each step. Allgower & Georg [**6**] describes two methods based on QR-factorization and LU-factorization of the Jacobian f_x^0, respectively, such that both the tangent vector and the Moore-Penrose inverse can easily be obtained from this factorization.

2. Linear solvers

The above mentioned approach using matrix factorizations is suitable for dense systems, but for large and sparse systems, iterative solvers should be considered. CONTBIF uses MATLAB's backslash operator '\', which uses the structure of $A = f_x^0$to decide on the best specific algorithm for solving the equation $Ax = b$, with respect to x. The systems that needs to be solved, for both equilibrium continuation and continuation of periodic solutions, have a special sparse structures which can be exploited to obtain more special and efficient solvers for the systems. In particular, AUTO uses bordering algorithms to solve the systems, see [**5**] and [**4**]. Such special solvers have not been implemented at the current stage of the program.

3. Calculating derivatives

All derivatives in CONTBIFare calculated by a central difference scheme

$$\frac{\partial f}{\partial x_i} = \frac{f(x + \epsilon e_i) - f(x - \epsilon e_i)}{2\epsilon}, \tag{71}$$

where ϵ is a user chosen constant and e_i is the vector $\begin{pmatrix} x_1 \\ \vdots \\ x_{n+1} \end{pmatrix}$ with

$$x_j = \begin{cases} 1 & \text{for } j = i \\ 0 & \text{otherwise} \end{cases}$$

.

For the numerical examples considered in this paper, continuation of equilibria works satisfactory with respect to computational time, but the computation of periodic solutions needs optimization. For example, since the structure of the Jacobian is known, see [**2**], we should avoid computing all the zero elements of the matrix. The computation time can also be reduced by replacing Newton's method by a dampened version. This means that we only evaluate the Jacobian after a certain number of iterations has been completed. Implemented in CONTBIF and the dampening constant is chosen by the user.

4. Other aspects regarding optimization

At the first 'naive' implementation of CONTBIF the Lagrange polynomials were constantly reevaluated, and by profiling the program it was noticed that considerable time was spent evaluating the Lagrange polynomials. However, given a mesh, the polynomials are not evaluated for other points than the collocation points. And as long as we keep the mesh fixed, the collocation points stay constant. Hence it suffices to evaluate the polynomials for each collocation point once, and storing the result in a vector. On example 4 in chapter 4, the cost of computing a branch of periodic solution were reduced by a factor of approximately 0.5. This in addition to not naively calculate Jacobian is discussed above reduced the computation time for a specific test example reduced the computation time from $38s$ to $5s$. If an adaptive mesh scheme as discussed in the previous chapter is selected there could also be some gain in evaluating the speed at each node and storing the result in a vector since it has to be calculated both by the monitor function and the phase condition.

CHAPTER 6

User Manual

The purpose of this chapter is to highlight the how the theory in the preceding chapters are implemented in CONTBIF and to be a user's guide to the program. For specific examples the reader is referred to the next chapter, and the scripts in the appendix.

1. General program overview

Before starting any scrips, the function *initcont()* must be invoked. The model problem must be specified in a separate MATLAB file according to the description given in 2. The basic sequence of CONTBIFis to run an *initializer* for the problem first and secondly run the *continuer*.

2. The input file

The input file shall have the following structure. Its return value must be a cell of two entries. The first entry are shall contain a pointer to a function, hereafter referred to as the *init function*. The purpose of the init function is to store problem specific parameters such as the initial point, and perhaps specific choices for parameters related to the continuation process. The second entry shall contain a pointer to the system that is to be solved, hereafter called the *vector field*. See table 1 for a summary.

3. Initializers

Three initializers are provided with the program; *initEP_EPcont* and *initBP_EPcont* which initializes equilibrium continuation from an equilibrium point or a branch point respectively, and *initH_LC* which initializes limit cycle continuation starting from a Hopf bifurcation point. The purpose of the initializers are to validate the initial points, compute an initial direction vector, and to set the global variables used by the continuer. Thus initializer must always be invoked before

TABLE 1. Input functions

Function	Input	Output
Init	Optional	Optional
Vector field	$[u, \lambda]$	$f(x, \lambda)$

starting the continuer. Basically the assumption is made that the continuation process starts near a regular point on the solution branch, and that bifurcation points detected by equilibrium continuation. Thus the syntax differs between the initializers. The initializer initEP_EPcont is called by *initEP_EPcont(funct,u,p,ap,opts)*, where *funct* is the input file, *u* is an approximation of an initial equilibria, *p* is a vector containing the parameters of the system, and *ap* is the index of the active parameter. The input argument *opts* are optional and is an option structure containing the options for the continuation process. See section 4 for further details on *opts*. In particular, note that the option *orientation*determines the direction of the initial predictor step, i.e if *orientation* $= +1$, the step is taken in the tangent direction, and vice versa. The syntax for starting an equilibrium computation of a new branch is done by calling; *initBP_EPcont(funct,sys,point,amp,opts)*, where *funct* and *opts* are as above, and *sys* is the output structure returned by the *continuer* (see section 5, and *point* is the index for the branch point in this structure. The initializer computes a second null vector and the first point on the new branch is predicted by taking a step of length *amp* in the direction of this null vector. The direction in which the new curve is traced is determined by the option *orientation*. Note that the concept of the tangent vector is not applicable at a bifurcation point, hence the sign of *orientation* simply determines if the curve is traced in the direction of the computed direction vector (whatever direction that may be). The initial direction, may however be inspected before starting the continuer by accessing the global variable *curve* (see section 6). Similarly a branch of periodic solutions may be started from a Hopf bifurcation point by calling; *initH_LCcont(funct,out,point,amp,opts)*. The argument are the same as for *initBP_EPcont*. Also note that the *orientation* is irrelevant in the context of computing a periodic solution starting at a Hopf point, since there is only one direction in which the periodic solutions persists.

For each type of initializers there are *model files* which defines the functions to be considered for the continuer, such as the test functions and corresponding labels and the system to be evaluated. They are named *EP*, *BP* and *LC* accordingly.

4. Options for the continuation process

Options are changed by creating a structure *opts* and setting $opts. < option >=< newvalue >$. The options that can be changed is listed in table 2. For some variables the symbol 1/0 is listed, the option is then a boolean indicating on/off, respectively. The variable *orientation* is special in this context since the valid arguments are ± 1.

5. The continuer

The *continuer* can be called without arguments, as the initial conditions are set by the initializers. However, it can take the arguments *funct* and *opts* (in this order), if the options are to be altered. The continuer returns a structure *out* containing the following information

- *out.x* contains the points of the solution branch. More precisely *out.x(:,i)* displays the point i, and the point looks like $[u, \lambda]^*$ for equilibrium continuation and $[u, T, \lambda]^*$ for continuation of periodic solutions.

TABLE 2. Options

Option	Description	Default value
h	initial steplength	10^{-3}
hmin	minimum steplength	10^{-6}
hmax	maximum steplength	$10^{-}3$
hinc	steplenght increase factor	1.3
hdec	steplength decrease factor	0.5
orientation	the orientation(±1) of the initial step	$+1$
secanttangent	secant prediction of the tangent 1/0	1
jacincr	perturbation for Jacobian calculations	10^{-5}
maxNewtIter	maximum number of newton iterations	20
detectBifurcationPoints	on/off 1/0	1
maxBifIter	maximum number of PC-steps for locating a bifurcation	20
toll	tolerance for new point $\frac{\lvert\Delta\lambda\rvert}{1+\lvert\lambda\rvert} < toll$	10^{-6}
tolu	tolerance for new point $\frac{\lVert\Delta u\rVert_\infty}{1+\lVert u\rVert_\infty} < tolu$	10^{-6}
tolxinf	tolerance for new point $\lVert \Delta x \rVert_\infty < tolxinf$	10^{-6}
tolfinf	tolerance for new point $\lVert f(x_1) - f(x_0) \rVert_\infty < tolfinf$	10^{-6}
tolx	tolerance for new point $\lVert \Delta x \rVert < tolx$	10^{-6}
tolf	tolerance for new point $\lVert f(x_i) - f(xi-1) \rVert < tolf$	10^{-6}
accepanceCriteria	vector indicating which of the above tolerances are active	$[1, 1, 0, 0, 0, 0]$
correctortype	0=gauss-newton, 1=pseudoarchlength	0
testtol	tolerance for accepting special points	10^{-5}
testsing	tolerance for accepting singular points	10^{-5}
ntst	Number of elements for the collocation method	4
ncol	Number of collocation points	3
printprogress	on/off 1/0, prints the progress of the continuation process	0
adaptmesh	on/off 1/0 mesh adaptation	0
smoothing	smoothing of monitor function on/off 1/0	0

- *out.v* contains the direction vector. The structure of this matrix is the same as for *out.x*.
- *out.s.index* contains the indices of the start point, the bifurcation points, and the end point.
- *out.s.label* contains the labels of the start point, bifurcation point, and end point.
- *out.mesh* contains the mesh if a mesh adaptation is active.

After a solution branch have been computed, the continuer can be started without an initializer in order to proceed further along the solution branch. However, when doing this the new output should be received by a different variable, as it does not incorporate the old solution into the new. Merging of solution structures may however be done by the function *mergecont*, which takes as input the two outputs *out1* and *out2* and merges them into one large output.

TABLE 3. Plotters

Property	Description
plotsys	(x,v,s,e*,f*)
plotperiodic	(x,n,e*)
plotpermaxmin2d	(out,coord)

TABLE 4. Input arguments

Property	Description
x	is the solution matrix (out.x)
v	is the matrix of direction vectors (out.v)
s	is the structure of special points (out.s)
e*	is a 2 or 3 dimensional vector defining which entries of x shall be plotted
f*	is the matrix of eigenvalues
n*	is the dimension of the ODE
out	is the entire output structure returned from the continuer
coord	is the coordinate that will be plotted against the active parameter

REMARK 41. *The user should access the global variable* curve *and set* curve.spstart = 0 *before continuing a continuation of periodic solutions.* Spstart *is a flag marking that the initial phase condition* (54) *is to be used, hence if we merely wish to continue an already existing one this flag should not be active. Also the output structure of the continuer has been made a global variable (out). This enables us to abort the continuer during the continuation process without loosing the already calculated data.*

6. Global variables

There are two global variables in use; *copt* and *curve*. They are both in the form of a structure. Generally the variable *copt* contains the options and references to the functions in the system file, while *curve* at any time contains the information needed by the continuer.

7. Plotting

The table 3 list the plotters and the table 4 lists the input arguments. Note that an (*) indicates that the input argument is optional. The plotter *plotsys* uses a full line for stable equilibria and a dotted line for unstable equilibria, while the function *plotpermaxmin2d* used small squares to denote limit cycles. The labels 'F', 'H' and 'B' are used to denote folds,Hopf bifurcations and branch points, respectively.

CHAPTER 7

Numerical examples

In this chapter we test the software on some examples. The purpose is to show that the program works well with regard to equilibrium continuation, and detection of bifurcation points, branch switching and to highlight some issues with regard to computation of periodic solution branches.

Example 1. We shall consider the dynamical system

$$\begin{aligned}\dot{x} &= -2x + y + \alpha e^{x}\\ \dot{y} &= e^{y}\alpha + x - 2y,\end{aligned}$$

starting at the inital point $x = y = \alpha = 0$. The system has a fold where the stability of the solution branch changes, as well as a branch point where two unstable solution branches meet. It may be of interest that the second branch cannot be stable as long as the first branch do not shift stability as it passes through the branch point. See [**6**] or [**9**] for further details of why this is so.

Example 2. We shall consider the dynamical system

$$\begin{aligned}\dot{x} &= \lambda x - y - x(x^2 + y^2)\\ \dot{y} &= x + \lambda y - y(x^2 + y^2)\end{aligned}$$

starting at the initial point $x = y = 0$, and $\lambda = -1$. The curious reader may note that this system is known as the *normal form* of the Hopf bifurcation. Thus it is a good example for testing the computation of periodic solution branches. The system has a Hopf bifurcation at $\lambda = 0$, where the stability of the equilibrium branch changes. The branch of periodic solutions emerging from the Hopf bifurcation point is stable, but since we have not implemented algorithms for obtaining the Floquet multipliers, this cannot be verified by the program.

Example 3. We shall consider the famous Lorenz system

$$\begin{aligned}\dot{x} &= \sigma(y - x)\\ \dot{y} &= \rho x - y - xz\\ \dot{z} &= xy - \beta z,\end{aligned}$$

starting at the inital point $x = y = z = 0$, $\sigma = 10$,$\beta = \frac{8}{3}$ and $\rho = 0$, using ρ as an active parameter. We see that the system is symmetric about the ρ-axis. The system contains a branch point and two Hopf bifurcation points, one of which a continuation of a solution branch of periodic solution has been computed. The bifurcation diagram is presented in figure 3 and the periodic solutions are shown in the subspace spanned by ρ, x and y in 4.

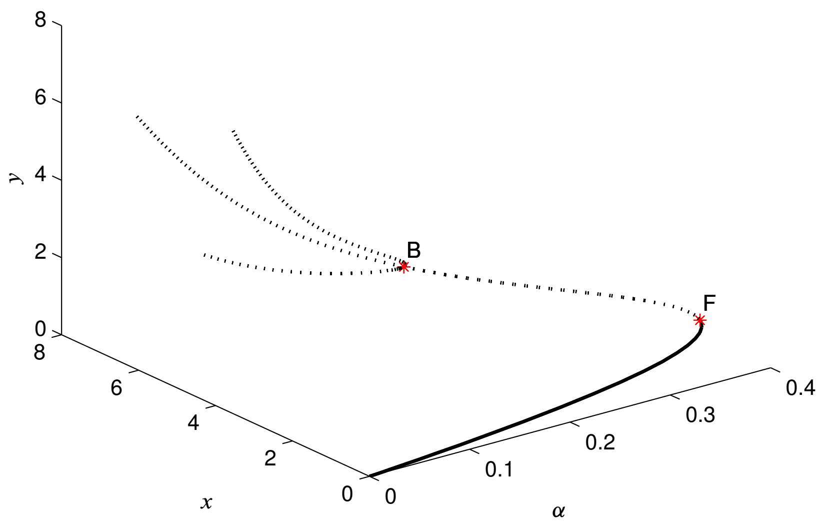


FIGURE 1. Example 1

Example 4. We shall consider the Van der Pol's equation;

$$\ddot{y} - (2\lambda - y^2)\dot{y} + y = 0,$$

which can also be written as a system of first order ordinary differential equations

$$\dot{u}_1 = u_2,$$
$$\dot{u}_2 = -u_1 + 2\lambda u_2 - u_1^2 u_2.$$

We shall set the start point to $u_1 = u_2 = 0$, and $\lambda = 1$, and start a continuation in the negative direction with respect to the tangent vector. There is a Hopf bifurcation point for $\lambda = 0$, and the figure 5 show the computed branch of periodic solutions originating at this point with an initial uniform mesh and $ntst = 10$, and $ncol = 4$. This is the first example that indicates the need of some mesh adaptation, as the nodes become unevenly distributed with respect to the arclength. This is also the case for the next example, in which we shall consider this in more detail.

Example 5. In this example we consider the Brusselator

$$\dot{x} = z(x, y, A, B) - x;$$
$$\dot{y} = -z(x, y, A, B);$$

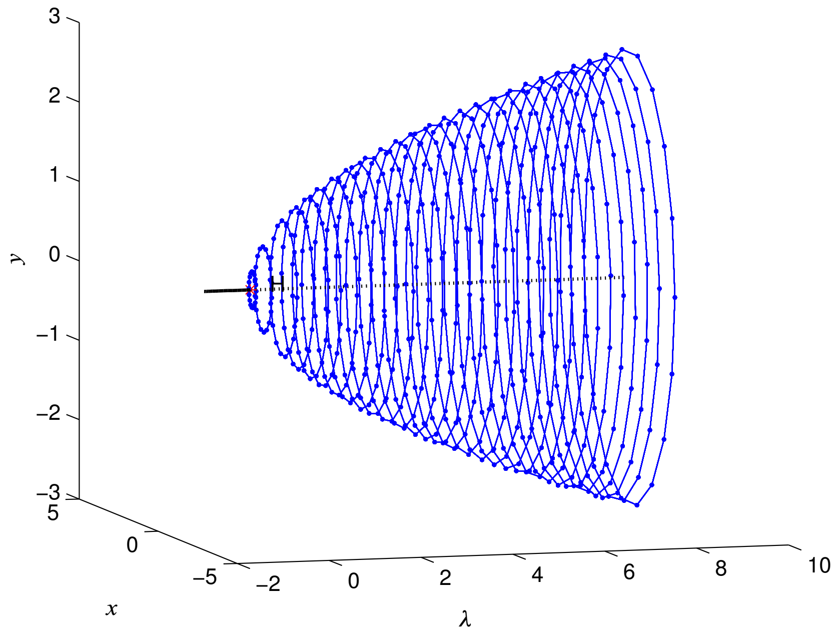


FIGURE 2. Example 2; Bifurcation diagram

where $z = Bx + A^2y + \frac{B}{A}x^2 + 2Axy + x^2y$. The initial point is set to $x = y = 0$ and $A = B = 1$, and B is set as the active parameter. The bifurcation diagram 6 shows the existence of a Hopf bifurcation point. The figures 7 and 8 shows the computation of the periodic solution branch with $ntst = 10$ and $ncol = 4$ on a fixed mesh. It is clearly visible on the figures that the nodes are unevenly distributed in space, and hence the solution looses accuracy. In order to improve on this we tested the mesh adaptation technique described in section 3 of chapter 4 for this particular problem. In general the performance was bad. The reason is that for this example a mesh that is closely equidistributed at one peridic solution does not translate well over to the next for reasonable small steplengths. The figures 9(a) and 9(b) shows the results for a small test example, with maximum steplength set to 0.1, $ntst = 8$ and $ncol = 4$. The two figures show the computed solutions for the same values of B except from the last periodic solution where the mesh did not converge and the steplength were reduced. The mesh tolerance was set to $\| t \|_\infty < 10^{-5}$. Similar results were also obtained for the Van der Pol system in example 4.

Example 6: We shall consider one example similar to those examples discussed in the project work [**14**].

$$u(x,t) = te^{-\frac{(x-a)^2}{2\epsilon^2}}.$$

This function represents a spike centered at a and where the thickness depends on ϵ. Note that the height of the spike at $t = t_0$ is t_0. The mesh as computed by $ode15i$ in MATLAB is shown

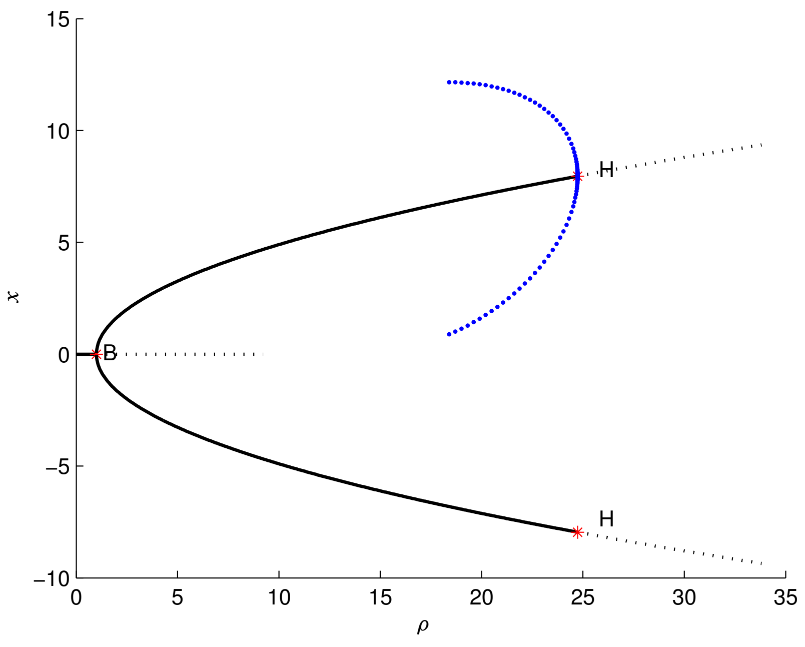


FIGURE 3. Example 3 Bifurcation diagram for the Lorenz system

TABLE 1. Parameters for example 5

Parameter	Value
τ	10^{-6}
ϵ	0.300145
a	0.0144
N	19

in figure 10 for the parameters listed in table 5. Note that N in this case refers to the number of internal nodes in the (moving) mesh, and τ is a relaxation constant for the MMPDE. In this case MMPDE4, that is equation (52) in [**14**], is used. A continuation of the periodic solution branch originating in the first Hopf bifurcation started for $ncol = 4$ and $ntst = 20$. The result is plotted for the node $x_1 0$ in 12. As is obvious from the figure, the branch did not get very far before the corrector failed ($\lambda = 0.083$).

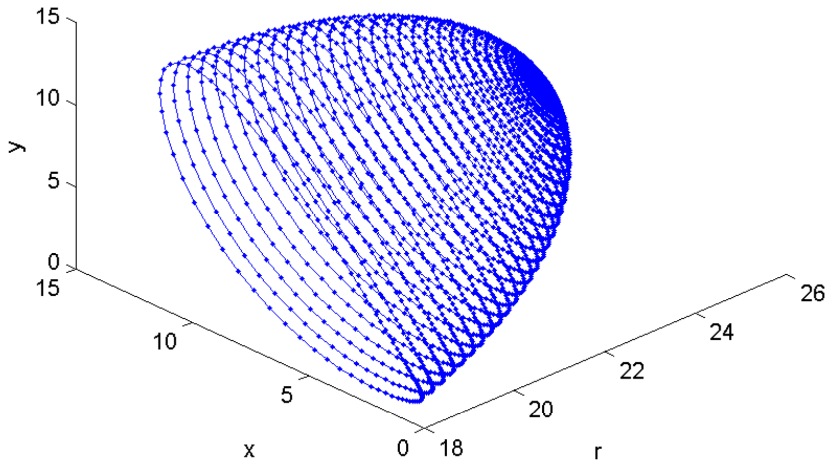


FIGURE 4. Example 3: Periodic solutions of the Lorenz system

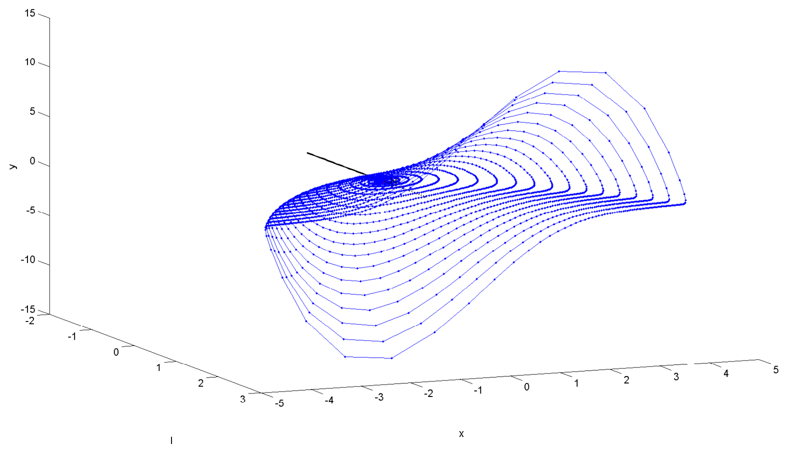


FIGURE 5. Example 4; Periodic solutions of the Van der Pol system

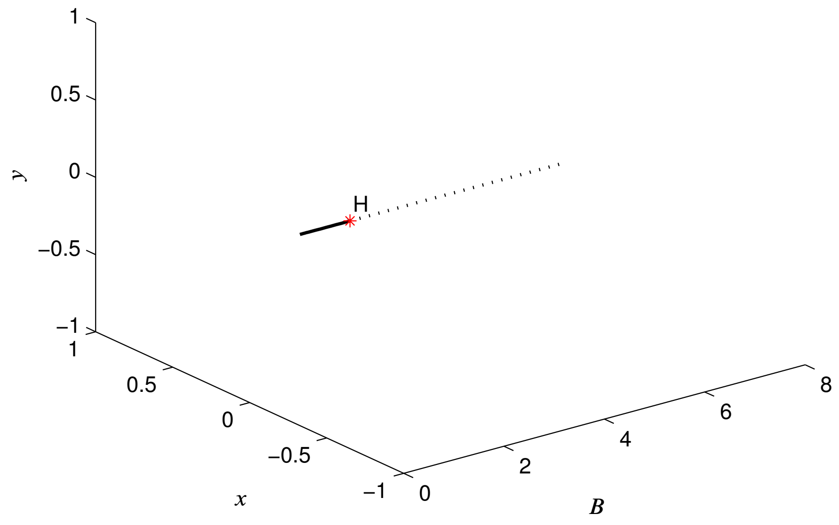


FIGURE 6. Example 5 Bifurcation diagram

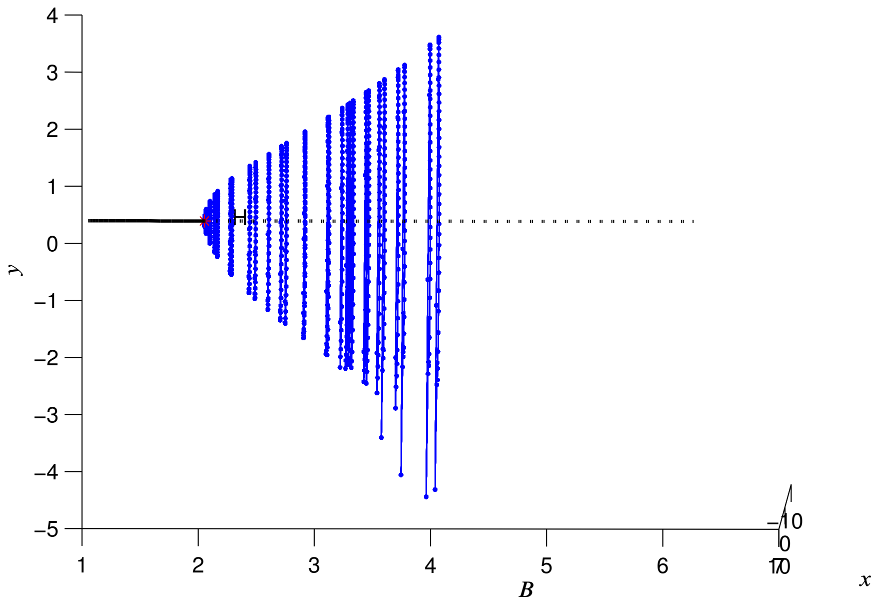


FIGURE 7. Example 5; without mesh adaptation

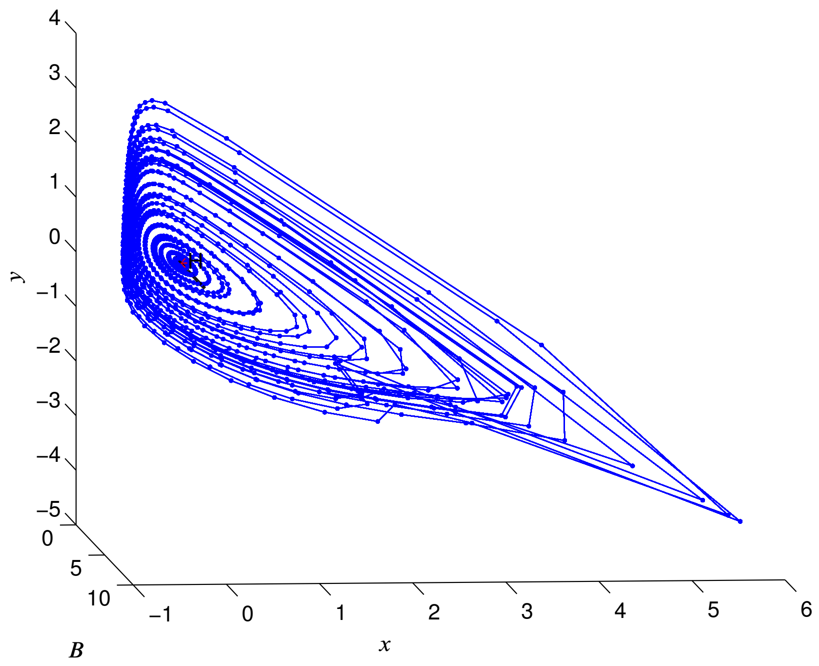
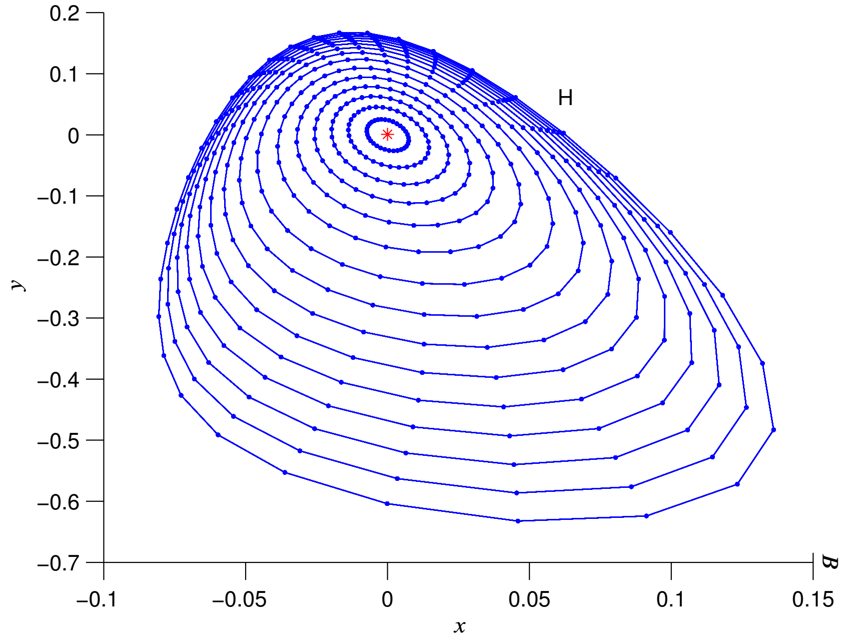
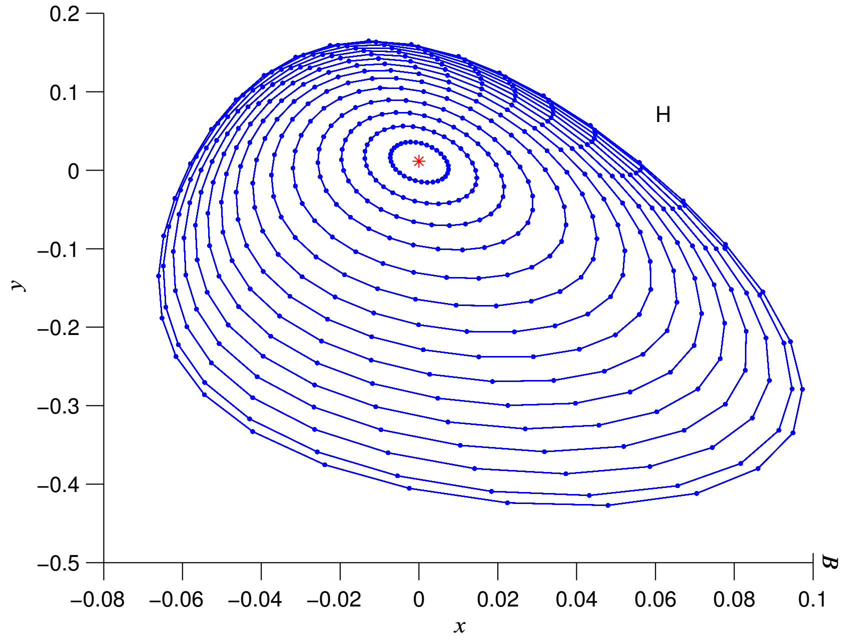


FIGURE 8. Example 5; alternate angle



(a) No mesh adaptation



(b) With mesh adaptation

FIGURE 9. Example 5; Test of mesh adaptation

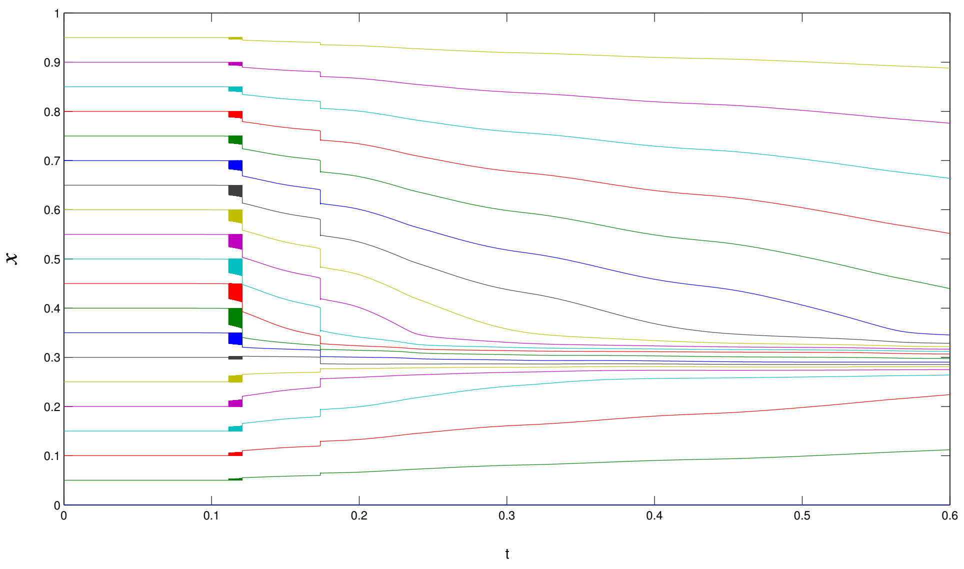


FIGURE 10. Example 5: Mesh

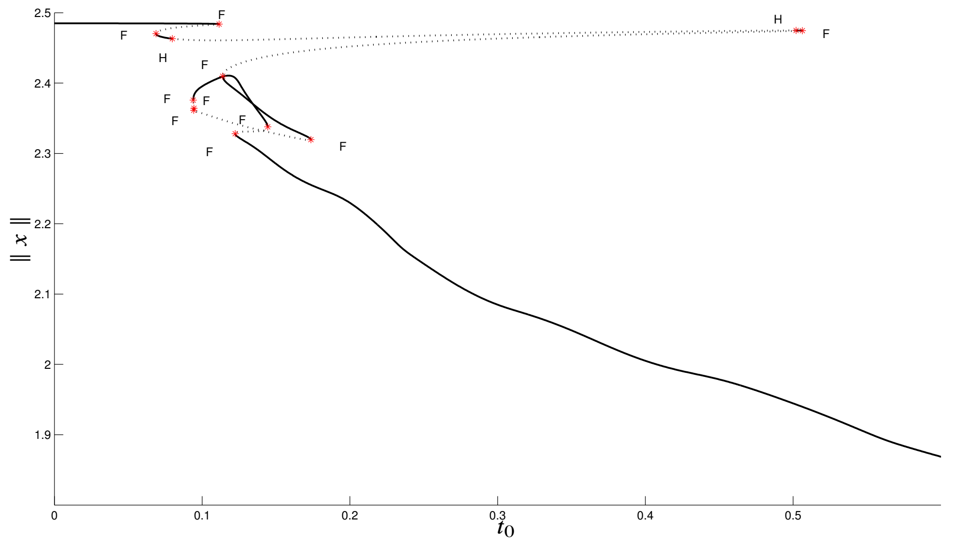


FIGURE 11. Example 5: Bifurcation diagram

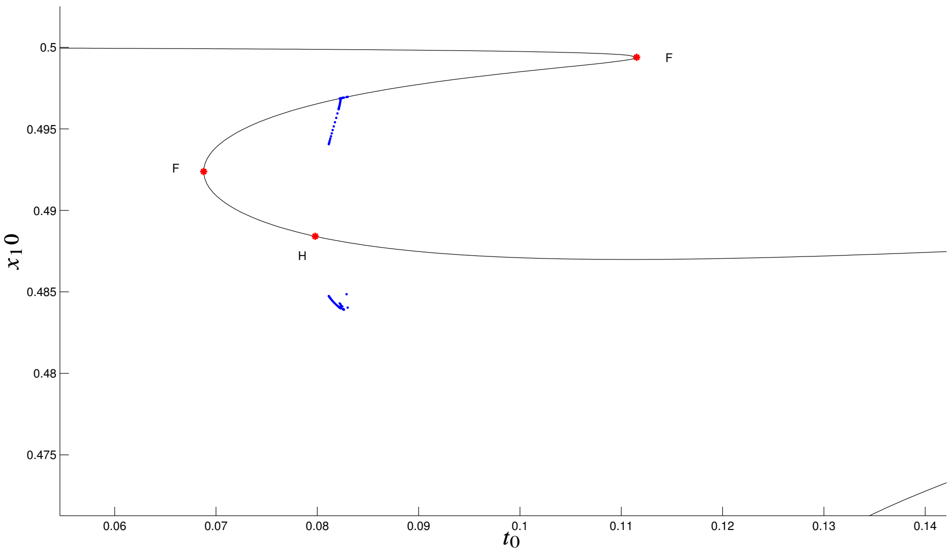


FIGURE 12. Example 5: Periodic solutions

CHAPTER 8

Conclusions

The numerical examples in the last chapter, shows that CONTBIF works fairly well for continuation of equilibria, detection of bifurcation points, and branch switching. In general for systems having regions of high density of bifurcation points relative to the steplength, there may be problems with detection of bifurcation points. Bifurcation points may not be detected or be wrongly identified if a step in the continuation process steps over more than one bifurcation point. Especially for large systems, some experimenting is needed to determine appropriate values for the maximum and minimum steplengths, in order to optimize with respect to the quality of the bifurcation diagram, and computation time. There is also an example where the branch switching and equilibrium continuation algorithms does fails to converge. This example is the predator-prey example

$$\dot{x} = 3x(1-x) - xy - \lambda(1-e^{-5x})$$
$$\dot{y} = -y + 3xy,$$

for which a bifurcation diagram have been computed by AUTO in [**2**]. Specifically the computation were started at the origin, with $\lambda = -5$. It may be that we would get better result by computing more accurate direction vector for the branch switching algorithm, but this have not been tested.

The continuation of periodic solution branches works well for the examples where the speed along the periodic solution is approximately constant at each periodic solution as a fixed uniform mesh is then evenly distributed in space. Although the attempts for mesh adaptation presented earlier did not work very well for the examples considered in this text, it would be interesting to study this in more detail. Another issue that should be considered is further optimization especially in the case of computation of periodic solution branches. Generally, it is our belief that algorithms for this in general should be compiled, and not interpreted as in this case. MATLAB offers the possibility of using mex-files, which essentially is C code, that may reduce the computation time tenfold. Other approaches is to implement specific linear solvers, and replace as many finite difference calculation by exact derivatives as possible. It would be interesting to see the impact of using the bordered method presented in [**4**] versus MATLABs backslash operator. It would also be interesting to be interesting to study how a conjugate gradient corrector would perform versus the Newton methods. In fact, the conjugate gradient solver as presented in [**6**] was implemented in CONTBIF, but was excluded from this project work since it did not provide significant improvements over the Newton methods for the test examples that were considered. Partly this was because we may expect that a larger number of corrector steps must be performed before the convergence criteria are met. The maximum number of corrector steps should therefore be modified accordingly, for otherwise the steplength algorithm will cause the steplength to decrease and hence increase the overall computation time as more points needs to be computed. However, the performance of conjugate gradient correctors would be worthwhile to study further. Another interesting prospect would be to implement a Fourier spectral method for computation of periodic solutions, this has the additional advantage that the Floquet multipliers are easily obtained. Moore [**12**] discuss the method, and argues that for the particular problem of computing periodic solution in a continuation process they do not require the solution of linear systems with non sparse coefficient matrices, which is generally considered the main disadvantage of the method. Finally, it would also be interesting to implement higher order predictors, to see if there is gain in either accuracy or in the overall computation time. Perhaps, higher order predictors will perform better than Euler prediction when periodic solutions are to be computed.

The general feeling at the end of this thesis is that we closer to the beginning than at the end of the project. There is much that is needed to be included in the program in order for it to be a fully fledged continuation software, and the problem of computing periodic solution branches still persists.

Bibliography

[1] Y.A. Kuznetsov W. Mestrom A.M Riet & B. Sautois A. Dhooge, W. Govaerts. Matcont and cl_matcont: Continuation toolboxes in matlab. December 2006.

[2] H.M. Osinga & J.Galan-Vioque B. Krauskopf. *Numerical Continuation Methods for Dynamical Systems*. Springer, first edition, 2007.

[3] E. Doedel. Numerical analysis of nonlinear equations, 2007.

[4] H.B. Keller & J.P. Kernevez E. Doedel. Numerical analysis and control of bifurcation problems. (i) bifurcation in finite dimensions. *International Journal of Bifurcation and Chaos*, 1(4):745–772, 1991.

[5] H.B. Keller & J.P. Kernevez E. Doedel. Numerical analysis and control of bifurcation problems. (i) bifurcation in finite dimensions. *International Journal of Bifurcation and Chaos*, 1(1):493–520, 1991.

[6] E.L Allgower & K. Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics (SIAM), first edition, 2003.

[7] J.Hale & H.Kocak. *Dynamics and Bifurcations*.

[8] Weizhang Huang, Robert, and D. Russell. Moving mesh partial differential equations (mmpdes) based on the equidistribution principle. *SIAM J. Numer. Anal*, 31:709–730, 1994.

[9] Y.A. Kuznetsov. *Elements of Applied Bifurcation Theory*. Springer, third edition, 2004.

[10] G.H. Goloub & C.F. Van Loan. *Matrix Computations*. The John Hopkins University Press, third edition, 1996.

[11] G. Moore. The numerical treatment of non-trivial bifurcation points. *Numerical Functional Analysis and Optimization.*, 2(6):441–472, 1980.

[12] G. Moore. Floquet theory as a computational tool. *Siam J. Numer. Anal.*, 42(6):2522–2568, 2005.

[13] L. Perko. *Differential Equations and Dynamical Systems*. Springer, 2000.

[14] E. Ravnås. Dynamical analysis of moving mesh partial differential equations, 2008.