Utrecht University

Bachelor thesis

Numerical continuation of closed invariant curves of maps and its implementation in the MATLAB software MatContM

Written by
Luuk Sterke

Under supervision of

Prof. Dr. Yuri A. Kuznetsov
Dr. Hil G.E. Meijer

Faculty of Science
Department of Mathematics
Utrecht University
June 2019

Abstract

This Bachelor Thesis aims at developing new methods for continuation of closed invariant curves of multi-dimensional diffeomorphisms and their actual implementation in MatContM. This software includes the standard predictor-corrector code to continue 1D solution branches implicitly defined by systems of algebraic equations. Thus, the main task is to formulate and implement a new defining system to continue closed invariant curves with constant rotation number in two control parameters. These defining equations are based on the Fourier approximation of the closed invariant curve and the discretization of the invariancy condition. An auxiliary but important subtask is to implement an initialization algorithm to start continuation of closed invariant curves from a given point in the Neimark-Sacker bifurcation curve.

I would like to extend my sincere thanks to:

Yuri Kuznetsov, for the help with understanding the theory in this thesis,

Hil Meijer, for the tremendous help with implementing and integrating these methods into MatContM.

Contents

Preface

Discrete-time dynamical systems generated by iterating diffeomorphisms exhibit various invariant sets, from fixed points and cycles to chaotic attractors. Closed invariant curves are examples of nontrivial invariant sets appearing in applications. It is well known that such an invariant curve can appear as a result of a Neimark-Sacker (NS) bifurcation, when the fixed point changes its linear stability via a pair of complex eigenvalues passing the unit circle.

There exist well known algorithms for numerical computation of fixed points and their bifurcations (including NS), which are implemented in the standard MATLAB bifurcation software MatContM. While some algorithms for continuation of closed invariant curves were proposed in the literature, they have not been implemented into any standard software for bifurcation analysis of diffeomorphisms.

This Bachelor Thesis aims at developing new methods for continuation of closed invariant curves of multi-dimensional diffeomorphisms and their actual implementation in MatContM. This software includes the standard predictor-corrector code to continue 1D solution branches implicitly defined by systems of algebraic equations. Thus, the main task is to formulate and implement a new defining system to continue closed invariant curves with constant rotation number in two control parameters. These defining equations are based on the Fourier approximation of the closed invariant curve and the discretization of the invariancy condition. An auxiliary but important subtask is to implement an initialization algorithm to start continuation of closed invariant curves from a given point in the NS bifurcation curve.

The Thesis includes an Introduction, where necessary mathematical notions and results will be presented, including the theory of the NS bifurcation. It also contain an explanation of the current functionality and continuation techniques implemented in MatContM software. Then, a detailed description of the new algorithms and their implementation in MATLAB is given. The new functionality is illustrated on two examples from applications.

Chapter 1

Introduction

1.1 Discrete time dynamical systems

Before we start, let us define a few basic notions and a system of which we want to study certain properties. In general, we will work with a map

$$x \mapsto f(x, \alpha), \tag{1.1}$$

where x is some vector in $\mathbb{R}^n$, α is a parameter vector in $\mathbb{R}^m$ and f is a smooth function. It will always be assumed that this map is a diffeomeorphism for each α, at least locally. We will use α, and later on a parameter β based on α, a lot in this thesis. Most statements, lemmas and theorems do not work for all α but only for sufficiently small $|\alpha|$. There is no indication of the limits of α, as these heavily depend on the concrete definition of f. We will not consider situations where $|\alpha|$ gets as large as to make our statements incorrect, as for these αs, the behaviour of our system is not the behaviour we want to study in this thesis.

Our first point of interest are the fixed points of f; points that remain unchanged under f. More formally, a point x_0 is called **fixed** if and only if $f(x_0, \alpha) = x_0$. The follow-up question to be asked is whether there are non-fixed points x_c for which there is a $K > 1$ in $\mathbb{N}$ such that $f^{(K)}(x_c, \alpha) = x_c$, where $f^{(K)}$ is the Kth iterate of (1.1). Such a point x_c is called **cyclic**, while the collection of all distinct images under f of a cyclic point is called a **cycle**.

Letting f act on a point x an arbitrary number of times, we obtain a (forward) **orbit**,

$$x, f(x, \alpha), f(f(x, \alpha), \alpha), f(f(f(x, \alpha), \alpha), \alpha), \ldots,$$

and have created a **discrete dynamic system**. Backward orbits can be defined similarly using the inverse of f. As we change the parameter α of this system, the topological properties of our system, such as number and stability of fixed and cyclic points, may change. In particular, at a certain value of α, let us denote it by α_0, fixed and cyclic points may appear or disappear. Such a moment in which the topological properties of a dynamical system change is what we call a **bifurcation**. Bifurcation theory has many applications. An elementary example from ecology is the scalar Ricker Map [8]

$$x \mapsto \alpha x \mathrm{e}^{-x}$$

that describes the year-to-year density of a fish population. This map has a trivial fixed point $x_0 = 0$ for all $\alpha > 0$. At $\alpha_1 = 1$, the trivial fixed point becomes unstable and a stable nontrivial fixed point $x_1(\alpha) = \ln \alpha$ appears. A cycle with period 2 splits from x_1 at $\alpha_2 = e^2 = 7.389\ldots$, when this fixed point also becomes unstable via the so called period-doubling bifurcation. Actually, an accumulating infinite sequence of period-doubling bifurcations happens in (1.10), leading to “chaos” at approximately $\alpha = 14.742\ldots$.

The main object of this thesis will be **closed invariant curves**. A “closed curve” is a homeomorphic (and often diffeomorphic) image of a circle. “Invariant” means that whenever we start at a point on such curve, we will always get to a point on the same curve again when we apply f. More formally, if curve C is in the domain of f, then C is an invariant curve if for any $x_0 \in C$, all elements in the forward orbit of x_0 are again in C. Clearly, closed invariant curves are possible for $n \geq 2$.

On this closed invariant curve we can also define the **rotation number** ρ, that characterizes the average angle by which the curve gets rotated if we let f act on it. To formally define the rotation number, let us parametrize the closed invariant curve C by an angle θ (mod 2π) and consider the map $\theta \mapsto \Psi(\theta)$ induced by the map f, i.e. $\Psi(\theta)$ is by definition the angle corresponding to $f(x, \alpha) \in C$ if θ corresponds to $x \in C$. We assume that Ψ preserves orientation on C.

$$\rho = \frac{1}{2\pi} \lim_{k \to \infty} \frac{\sum_{j=1}^{k} (\Psi^j(\theta) - \Psi^{j-1}(\theta))}{k}.$$

Clearly, if we got a rational rotation number $\rho = \frac{p}{q}$, where $p, q \in \mathbb{N}$, then we got cycles in C

$$x, f(x, \alpha), \ldots f^K(x, \alpha) = x$$

of at most length q. The dynamics of smooth maps with irrational rotation numbers is described by the following classical result.

Theorem 1.1.1 (Denjoy, [1])**.** *If Ψ is at least twice differntiable and ρ is irrational, then Ψ is topologically equivalent to a rigid rotation through the angle $2\pi\rho$, i.e. the map $\theta \mapsto \theta + 2\pi\rho$ (mod 2).*

1.2 Orbit structure near a fixed point

In this thesis, we will encounter a specific kind of bifurcation: a Neimark-Sacker bifurcation of a fixed point. We start with a discrete dynamical system as described in Paragraph 1.1. As f is a function from $\mathbb{R}^{n+m}$ to $\mathbb{R}^n$, the Jacobian matrix

$$A = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix} \tag{1.2}$$

is an $n \times n$ matrix, in general depending on parameter α. Suppose that at some fixed α_0 the map f has a fixed point x_0. Denote by $\mu_1, \mu_2, \ldots, \mu_n$ the eigenvalues of A. Suppose that there are n_s stable eigenvalues with $|\mu| < 1$, n_c critical eigenvalues with $|\mu| = 1$, and n_u unstable eigenvalues with $|\mu > 1$.

Definition 1.2.1. A fixed point is called **hyperbolic** if $n_c = 0$.

Theorem 1.2.1 (Grobman–Hartman, [3])**.** *Near a hyperbolic fixed point $x_0 = 0$, any diffeomorphism $x \mapsto f(x, \alpha_0)$ is locally topologically equivalent to the linear map $x \mapsto Ax$.*

This theorem implies that near a hyperbolic fixed point x_0 the qualitative properties of f, specifically stability of x_0 and asymptotic behaviour of orbits near x_0, will be the same as of the map

$$x \mapsto Ax. \tag{1.3}$$

This means that we can expect bifurcations of a fixed point x_0 close to our initial parameter value $\alpha = \alpha_0$, when A has one or more eigenvalues with $|\mu| = 1$. Given a critical parameter value $\alpha = \alpha_0$, simplest bifurcations of non-hyperbolic fixed points can be categorized into three groups:

- Period-doubling bifurcations [5, pp. 119–124], which occur if A has an eigenvalue $\mu = -1$,
- Fold bifurcations [10], which occur if A has an eigenvalue $\mu = 1$ and

- Neimark-Sacker bifurcations [5, pp. 129–137], which occur if A has eigenvalues $\mu_{1,2} = \mathrm{e}^{\pm i\theta_0}$.

The dynamics near a non-hyperbolic fixed point is more complicated and is described by the following theorem. Write the diffeomorphism (1.3) in its eigenbasis:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \mapsto \begin{pmatrix} A_c u + G(u,v,w) \\ A_u v + U(u,v,w) \\ A_s w + S(u,v,w) \end{pmatrix}, \tag{1.4}$$

where $(u,v,w) \in \mathbb{R}^{n_c} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_s}$, so that matrix A_c has n_c eigenvalues with $|\mu| = 1$, while A_u and A_s have only unstable and stable eigenvalues, respectively. Here G, U, S begin with nonlinear terms.

We then apply the Center Manifold Theorem for n dimensional maps.

Theorem 1.2.2 (Center Manifold Theorem, [9])**.** *If* $(u_, v_0, w_0) = 0$ *is a non-hyperbolic fixed point of* (1.4) *then there exists a local smooth invariant* n_c*-dimensional* **center manifold**

$$W^c = \{(u,v,w) \in \mathbb{R}_n : (v,w) = H(u)\}$$

where $H : \mathbb{R}^{n_c} \to \mathbb{R}^{n_u} \times \mathbb{R}^{n_s}$ *is such that* $H(u) = \mathcal{O}(\|u\|^2)$.

Moreover, the diffeomorphism (1.4) *is locally topologically equivalent near the origin to*

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \mapsto \begin{pmatrix} A_c u + g(u) \\ A_u v \\ A_s w \end{pmatrix},$$

where $g(u) := G(u, H(u))$.

Though the dynamics on the center manifold near the non-hyperbolic point do depend on nonlinear terms, the dynamics in the stable and unstable eigenspaces are still equivalent to those of linear maps.

1.3 Neimark-Sacker bifurcations

Before we can analyze Neimark-Sacker bifurcations, we will need to find a normal form of f, a simplification of f that does not change f's local qualitative properties. We will formulate such a normal form based on [5], in which some of the proofs of the results we need can also be found.

1.3.1 Planar case

Matrix $A(\alpha)$ in which we vary the parameter $\alpha \in \mathbb{R}$ has two eigenvalues, depending on this α, denoted by $\mu_{1,2}(\alpha) = r(\alpha)\mathrm{e}^{\pm i\theta(\alpha)}$. From our initial eigenvalues $\mu_{1,2} = \mathrm{e}^{\pm i\theta_0}$, we find that in $\mu(\alpha)$ we must have $r(\alpha_0) = 1$ and $\theta(\alpha_0) = \theta_0$.
Now we introduce a smooth function $\beta(\alpha)$, so that we can substitute $r(\alpha)$ by $1 + \beta(\alpha)$, such that $\beta(\alpha) = \beta(\alpha_0) = 0$ and suppose that $\beta'(\alpha_0) \neq 0$, so we can use β as our new unfolding parameter. Then we can rewrite our eigenvalues as $\mu_1(\beta) = \mu(\beta) = (1+\beta)\mathrm{e}^{i\phi(\beta)}$ and $\mu_2(\beta) = \overline{\mu}(\beta)$, the complex conjugate of $\mu_1(\beta)$. Now we will rewrite our map as a map of one complex variable,

$$z \mapsto \mu(\beta)z + g(z, \overline{z}, \beta), \tag{1.5}$$

with a function g we will find now.
First, let $q(\beta)$ and $p(\beta)$ be eigenvectors of $A(\beta)$ and $A^T(\beta)$, the transposed of matrix $A(\beta)$, respectively. Then we normalize $p(\beta)$ with respect to $q(\beta)$, so we have $\langle p(\beta), q(\beta)\rangle = 1$. Here $\langle p(\beta), q(\beta)\rangle = \overline{p}(\beta)^T q(\beta)$ is the standard complex vector inner product.
We can now uniquely define the variable z in (1.5), since any $x \in \mathbb{R}^2$ can be written as $zq(\beta) + \overline{z}\overline{q}(\beta)$

for some $z \in C$. We extract z from this equation by taking the inner product with $p(\beta)$ and then using basic rules for the inner product. This gives us the expression

$$z = \langle p(\beta), x \rangle.$$

Now, using (1.11) and substituting x, we see that the system defined in (1.5) is equivalent to

$$z \mapsto \mu(\beta)z + \langle p(\beta), F(zq(\beta) + \overline{zq}(\beta), \beta)\rangle.$$

From this we conclude that

$$g(z, \overline{z}, \beta) = F(zq(\beta) + \overline{zq}(\beta), \beta)\rangle.$$

Considering that we know F is the Taylor expansion of f, we can further define $g(z, \overline{z}, \beta)$ as

$$g(z, \overline{z}, \beta) = \sum_{k+l\geq 2} \frac{1}{k!l!} g_{kl}(\beta) z^k \overline{z}^l,$$

where g_{kl} denotes the lth partial derivative of g with respect to z followed by the kth partial derivative with respect to $\overline{z}$.

In the following lemmas we will omit parameter β, writing $\mu = \mu(\beta)$ and $g_{kl} = g_{kl}(\beta)$.

Lemma 1.3.1. *If $e^{i\theta_0} \neq 1$ and $e^{3i\theta_0} \neq 1$, we can transform the map*

$$z \mapsto \mu z + \frac{g_{02}}{2} z^2 + g_{11} z\overline{z} + \frac{g_{20}}{2} \overline{z}^2 + \mathcal{O}(|z|^3),$$

into the map

$$w \mapsto \mu w + \mathcal{O}(|w|^3),$$

using an invertible change of coordinates

$$z = w + \frac{h_{20}}{2} w^2 + h_{11} w\overline{w} + \frac{h_{02}}{2} \overline{w}^2,$$

where

$$h_{20} = \frac{g_{20}}{\mu^2 - \mu}, h_{11} = \frac{g_{11}}{|\mu|^2 - \mu}, h_{02} = \frac{g_{02}}{|\mu|^2}.$$

Lemma 1.3.2. *If $e^{2i\theta_0} \neq 1$ and $e^{4i\theta_0} \neq 1$, we can transform the map*

$$z \mapsto \mu z + \frac{g_{30}}{6} z^3 + \frac{g_{21}}{2} z^2\overline{z} + \frac{g_{12}}{2} z\overline{z}^2 + \frac{g_{03}}{6} \overline{z}^3 + \mathcal{O}(|z|^4)$$

into the map

$$w \mapsto \mu w + c_1 w^2\overline{w} + \mathcal{O}(|w|^4),$$

using an invertible change of coordinates

$$z = w + \frac{h_{30}}{6} w^3 + \frac{h_{21}}{2} w^2\overline{w} + \frac{h_{12}}{2} w\overline{w}^2 + \frac{h_{03}}{6} \overline{w}^3,$$

where

$$h_{30} = \frac{g_{30}}{\mu^3 - \mu}, h_{12} = \frac{g_{12}}{\overline{\mu}|\mu|^2 - \mu}, h_{03} = \frac{g_{03}}{\overline{\mu}^3 - \mu}..$$

Note that we did not mention an expression for h_{21}. This is because the expression we would like to use for h_{21},

$$h_{21} = \frac{g_{21}}{\mu(1 - |\mu|^2)},$$

is not smooth around $\beta = 0$. Indeed, as β approaches zero, $\mu(\beta)$ approaches one, resulting in h_{21} diverging to $\pm\infty$. Therefore we will set $h_{21} = 0$, which gives

$$c_1 = \frac{g_{21}}{2}.$$

Now we combine these results. With the same definitions as before, we get a transformation of our original map, formulated slightly differently.

Lemma 1.3.3. *The map*

$$z \mapsto \mu z + \frac{g_{20}}{2} z^2 + g_{11} z\overline{z} + \frac{g_{02}}{2}\overline{z}^2 + \frac{g_{30}}{6} z^3 + \frac{g_{21}}{2} z^2\overline{z} + \frac{g_{21}}{2} z\overline{z}^2 + \frac{g_{03}}{6}\overline{z}^3 + \mathcal{O}(|z|^4)$$

can be transformed into the map

$$w \mapsto \mu w + c_1 w^2 \overline{w} + \mathcal{O}(|w|^4)$$

using a smooth invertible transformation

$$z = w + \frac{h_{20}}{2} w^2 + h_{11} w\overline{w} + \frac{h_{02}}{2}\overline{w}^2 + \frac{h_{30}}{6} w^3 + \frac{h_{12}}{2} w\overline{w}^2 + \frac{h_{03}}{6}\overline{w}^3$$

with suitable $h_{jk},\ j,k \geq 0,\ 2 \leq j+k \leq 3.$

We need to recompute c_1 for the new coordinate system, using the Taylor coefficients of the original map, resulting in

$$c_1 = \frac{g_{20} g_{11}(\overline{\mu} - 3 + 2\mu)}{2(\mu^2 - \mu)(\overline{\mu} - 1)} + \frac{|g_{11}|^2}{1 - \overline{\mu}} + \frac{|g_{02}|^2}{2(\mu^2 - \overline{\mu})} + \frac{g_{21}}{2}, \tag{1.6}$$

which is again a function of β. At the bifurcation parameter value, $\beta(0) = 0$ and

$$c_1(0) = \frac{g_{20}(0) g_{11}(0)(1 - 2\mu_0)}{2(\mu_0^2 - \mu_0)} + \frac{|g_{11}(0)|^2}{1 - \bar{\mu}_0} + \frac{|g_{02}(0)|^2}{2(\mu_0^2 - \bar{\mu}_0)} + \frac{g_{21}(0)}{2}, \tag{1.7}$$

where $\mu_0 = \mathrm{e}^{i\theta_0}$.

Let us now first consider the planar case.

Theorem 1.3.4. *Consider a two dimensional discrete time system*

$$x \mapsto f(x, \alpha), \quad x \in \mathbb{R}^2, \alpha \in \mathbb{R}, \tag{1.8}$$

where

- *f is a smooth function,*
- *for small enough $|\alpha|$, f has a fixed point at $x = 0$.*
- *at this fixed point, the Jacobian matrix of f has eigenvalues $\mu_{1,2}(\alpha) = r(\alpha)e^{i\phi(\alpha)}$, where $r(0) = 1$ and $\phi(0) = \theta_0$,*
- *$r'(0) \neq 0$,*
- *$e^{ik\theta_0} \neq 1$ for $k = 1, 2, 3, 4$, which is equivalent to $\theta_0 \notin \{0, \frac{\pi}{2}, \pi, \frac{2\pi}{3}\}$.*

Then, by smooth invertible coordinate and parameter changes, we can transform the map (1.8) *into*

$$z \mapsto (1+\beta)e^{i\theta(\beta)}z + c_1(\beta)z|z|^2 + \mathcal{O}(|z|^4), \quad z \in \mathbb{C}, \beta \in \mathbb{R}, \tag{1.9}$$

which is equivalent to

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \mapsto (1+\beta) \begin{bmatrix} \cos\theta(\beta) & -\sin\theta(\beta) \\ \sin\theta(\beta) & \cos\theta(\beta) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + (y_1^2+y_2^2) \begin{bmatrix} \cos\theta(\beta) & -\sin\theta(\beta) \\ \sin\theta(\beta) & \cos\theta(\beta) \end{bmatrix} \begin{bmatrix} a(\beta) & -b(\beta) \\ b(\beta) & a(\beta) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \mathcal{O}(\|y\|^4), \tag{1.10}$$

where $z = y_1 + iy_2$ *and* $\|y\|$ *is the standard Euclidian norm,* $\sqrt{\langle y, y\rangle}$*,* $\theta(0) = \theta_0$*,* c_1 *is the function from formula* (1.6) *and* $a(\beta)$ *and* $b(\beta)$ *are real-valued functions such that* $a(\beta) + ib(\beta) = e^{-i\theta(\beta)}c_1(\beta)$*.*

It is easy to see that map (1.10) without the $\mathcal{O}(\|y\|^4)$-terms has a unique closed invariant curve, provided $a(0) \neq 0$. Then we get to perhaps one of the most important theorems in this thesis:

Theorem 1.3.5 (Generic Neimark-Sacker bifurcation)**.** *Consider a map* (1.8) *that satisfies all conditions of* Theorem 1.3.4*, as well as* $a(0) \neq 0$*. Then there exists a neighbourhood of* $x_0 = 0$ *in which a unique closed invariant curve bifurcates from* x_0 *as* α *passes through* $\alpha = 0$*.*

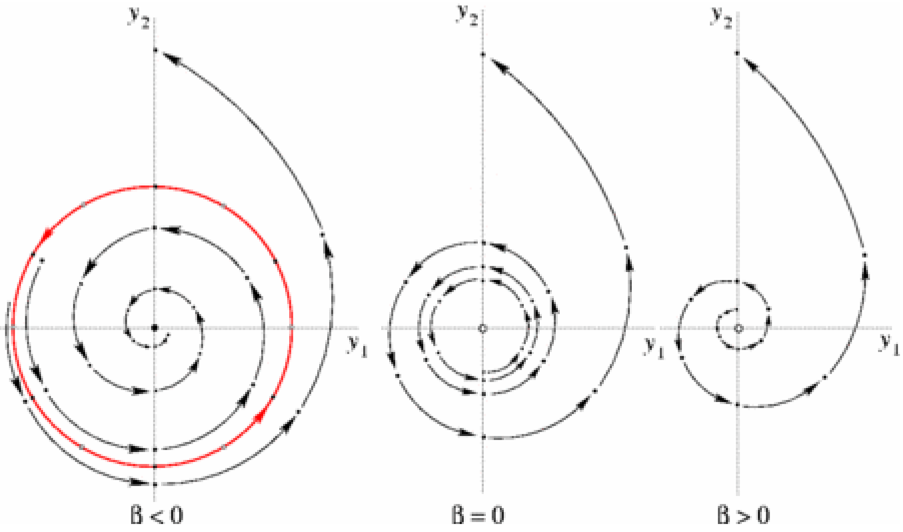

Figure 1.1: Here we see the subcritical case. The red circle represents a repelling invariant curve.

Definition 1.3.1. The real number

$$a(0) := \mathrm{Re}(e^{-i\theta_0}c_1(0))$$

is called the **first Lyapunov coefficient** for the Neimark-Sacker bifurcation.

The smoothness of the closed invariant curve appearing at the NS-bifurcation according to Theorem 1.3.5 is in general finite but increases when α approaches the critical parameter value $\alpha = 0$.

1.3.2 Multidimensional case

As A is the matrix representing the first order terms in f's Taylor expansion, we can write our system around the given fixed point x_0 as

$$f(x_0 + x, \alpha_0) = x_0 + A(\alpha_0)x + \frac{1}{2}B(\alpha_0; x, x) + \frac{1}{6}C(\alpha_0, x, x, x) + \mathcal{O}(\|x\|^4). \tag{1.11}$$

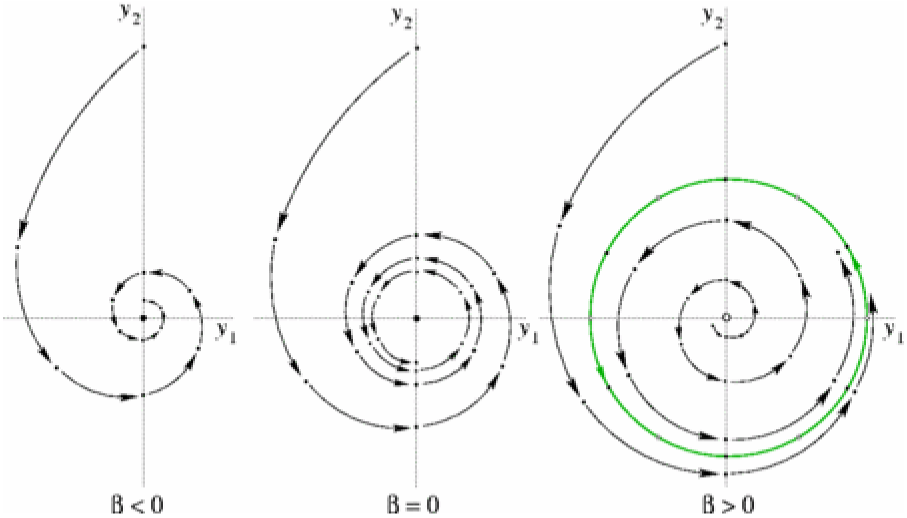

Figure 1.2: Here we see the supercritical case. The green circle represents an attracting invariant curve.

Here B and C are multi-linear vector functions of dimension n with the ith coordinates

$$\begin{aligned} B_i(\alpha_0; x, v) &= \sum_{j,k=1}^{n} \left. \frac{\partial^2 f_i(\xi, \alpha_0)}{\partial \xi_j \partial \xi_k} \right|_{\xi = x_0} x_j v_k, \\ C_i(\alpha_0; x, v, w) &= \sum_{j,k,l=1}^{n} \left. \frac{\partial^3 f_i(\xi, \alpha_0)}{\partial \xi_j \partial \xi_k \partial \xi_l} \right|_{\xi = x_0} x_j v_k w_l \ . \end{aligned}$$

Let $\mu_1, \mu_2, \ldots, \mu_n$ again be the eigenvalues of the Jacobian matrix, depending on α. As before, assume that there are n_c eigenvalues with $|\mu| = 1$, n_s eigenvalues with $|\mu| < 1$ and n_u eigenvalues with $|\mu| > 1$. Now we assume there is a fixed point $x_0 = 0$ at $\alpha_0 = 0$, which we could force by translation, if necessary. Introduce a new map

$$\begin{pmatrix} \alpha \\ x \end{pmatrix} \mapsto \begin{pmatrix} \alpha \\ f(x, \alpha) \end{pmatrix}, \tag{1.12}$$

where $\alpha \in \mathbb{R}^m$. Then at the fixed point $(x_0, \alpha_0) = (0, 0)$ we get the Jacobian matrix

$$J = \begin{bmatrix} I_m & 0 \\ f_\alpha(0,0) & f_x(0,0) \end{bmatrix},$$

where I_m is the unit $m \times m$ matrix, $f_\alpha(0,0)$ is an $m \times n$ matrix with on rows the partial derivatives of f with respect to α_j, where j is the column number as well as the number of the component of α with respect to which we take a partial derivative and $f_x(0,0)$ is our matrix A from (1.2). Matrix J has $(n_c + m)$ eigenvalues with absolute value 1 and $(n - n_c)$ eigenvalues with a nonzero $|(\mu| - 1)$. Now let the space with a basis of (generalized) eigenvectors corresponding to the $(n_c + m)$ eigenvalues on the unit circle be denoted by T. Now by the Center Manifold Theorem 1.2.2, there is a locally defined smooth $(n_c + m)$-dimensional manifold W, that is tangent to T in x_0. Now as α remains unchanged by (1.12), every set of the form $\{(\alpha, x) : \alpha = \alpha_0\}$ is invariant with respect to this map. This means that for each α we get a separate invariant manifold W_α of dimension n_c. We also call W_α a **center manifold**.

We will now introduce some local coordinates $u \in \mathbb{R}^{n_c}$ on W_α and look at the restriction of (1.12) to W_α

$$u \mapsto \Phi(u, \alpha), \tag{1.13}$$

which is again a smooth map. At $\alpha = 0$ it has a the fixed point $u = 0$ with n_c eigenvalues with absolute value 1.

We now apply these results to the Neimark-Sacker bifurcation in the n-dimensional case. Here the matrix $A(0)$ has $n_c = 2$ critical eigenvalues, so that map (1.13) is two-dimensional and exhibits a planar Neimark-Sacker bifurcation discussed above. We know that there is a complex coordinate in which (1.13) can be written as

$$w \mapsto \Phi_\beta(w) = e^{i\theta(\beta)}(1+\beta)w + c_1(\beta)w|w|^2 + \mathcal{O}(|w|^4)$$

with β a smooth function of α as in the planar case, $\theta(0) = \theta_0$, and $c_1(\beta)$ the same function as in (1.6). One can prove (see [5, 7]) that

$$c_1(0) = \frac{1}{2}\langle p, C(q,q,\overline{q}) + 2B(q,(I_n - A(0))^{-1}B(q,\overline{q})) + B(\overline{q},(e^{2i\theta_0}I)n - A(0))^{-1}B(q,q)))\rangle ,$$

where $q, p \in \mathbb{C}^n$ are such that

$$A(0)q = e^{i\theta_0}q, \quad A^T(0)p = e^{-i\theta_0}p$$

and $\langle q, q\rangle = \langle p, q\rangle = 1$. From the first Lyapunov coefficient $a(0) = \mathrm{Re}(e^{-i\theta_0}c_1(0))$, we can draw conclusions about dynamics on the the center manifold exactly as in the planar case, see Figures 1.1 and 1.2. The stability of the center manifold itself depends on the numbers n_s and n_u.

1.4 Numerical continuation of curves

The numerical continuation of a curve here is the approximation of a curve by using the predictor-corrector technique [4, 7]. In short the idea is to start in a point X^0 and use a **defining function** F to guess the next point, X^1, using the derivative of F in X^0. Then we correct this prediction by solving systems of equations. This process is known as the Moore-Penrose algorithm. Generally, we want to formulate our initial problem as a suitable algebraic problem

$$F(x) = 0,$$

where the defining function F is a function from $\mathbb{R}^M$ to $\mathbb{R}^{M-1}$, with M depending on the problem. F is also known as the **defining system**. For the continuation, both F and its partial derivatives are being used. When the initial point $x^{(0)}$ on the curve has been found, that means that $x^{(0)}$ satisfies $F(x^{(0)}) = 0$, we predict what the next point on a curve will be by taking a small step into the direction of the curve at $x^{(0)}$. Lets say this tangent vector is $v^{(0)}$, then we predict that our next point will be

$$X^0 = x^{(0)} + hv^{(0)}.$$

In general, X^0 will not be the correct next point on the curve, so we need to correct X^0 in order to get the next point on the curve $x^{(1)}$. As we know that $x^{(1)}$ has to be on the curve and for small h it has to be in this direction somewhere, we will try to find $x^{(1)}$ by solving an optimization problem using our approximation and knowledge about F. We get

$$x^{(1)} = \min_{x \in \mathbb{R}^{N+1}} \{\|x - X^0\| | F(x) = 0\}.$$

Finding this minimum is equivalent to computing X satisfying

$$\begin{cases} F(X) = 0, \\ \langle v, X - X^0\rangle = 0, \end{cases}$$

where v is the tangent vector to the curve $F(x) = 0$ at X. Using the linear approximation, we replace this system by

$$\begin{cases} F(X^0) + DF(X^0)(X^1 - X^0) = 0, \\ \langle V^0, X^1 - X^0 \rangle = 0, \end{cases}$$

where V^0 is the normalized tangent vector to $F(X) = F(X^0)$ at X^0, i.e.

$$DF(X^0)V^0 = 0, \quad \|V^0\| = 1,$$

and where X^1 is our next best approximation for $x^{(1)}$ (see Figure 1.3(a)). However, X^1 may not yet be close enough to the real $x^{(1)}$ and we may want to get more precision. In this case, we repeat part of the process. Now we will have the normalized tangent vector V^1 of $F(X) = F(X^1)$ in the point X^1 such that

$$DF(X^1)V^1 = 0, \quad \|V^1\| = 1,$$

and try to find a better approximation X^2 by solving the new system

$$\begin{cases} F(X^1) + DF(X^1)(X^2 - X^1) = 0 \\ \langle V^1, X^2 - X^1 \rangle = 0. \end{cases}$$

With every such step we will get closer to some point $x^{(1)}$ in the curve. If after k such steps both the correction $\|X^k - X^{k-1}\|$ and the error $\|F(X^k)\|$ are smaller than some margin of error ϵ, we take $x^{(1)} = X^k$ and we repeat the whole process to find the next point on the curve, $x^{(2)}$. Actually, the algorithm implemented in MatContM is slightly different (see Figure 1.3(b)), but the idea is the same.

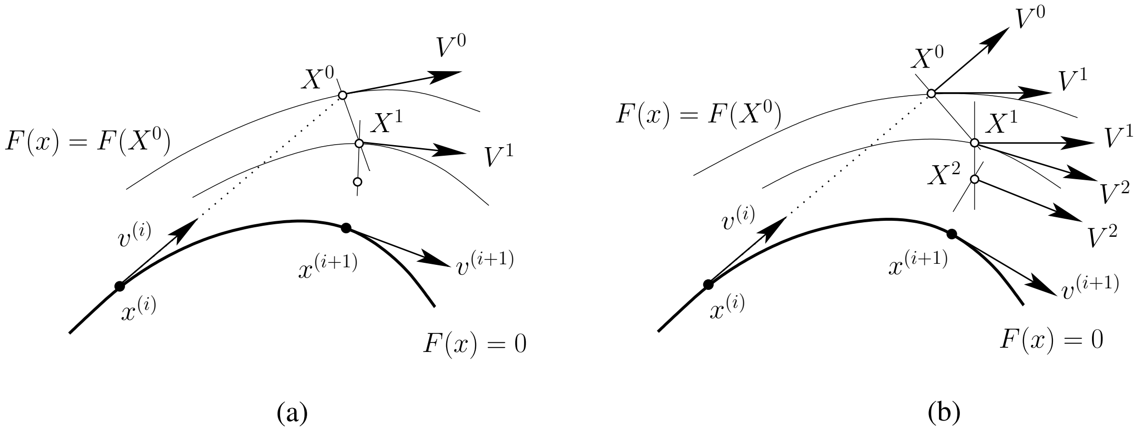

Figure 1.3: Numerical continuation: (a) Moore–Penrose algorithm; (b) its simplified implementation in MatContM.

If we use a small step size h, this numerical continuation of the curve will approach the exact curve, if we connect the points $x^{(0)}, x^{(1)}, \ldots, x^{(L)}$ for some L to which we wish to continue the process.

1.5 MatContM

For the numerical continuation of points, fixed points and curves, there is a piece of software created in the domain specific language of MATLAB, that goes by the name of MatContM [4]. The software is able to provide its users with:

- Simulation and visualization of maps as described above.

- Computation of the Lyapunov exponent at any point in the orbit of a map.
- Continuation of fixed points of maps depending on the given control parameter, such as our α.
- Detection of bifurcations such as the Neimark-Sacker bifurcation, but also fold and flip bifurcations.
- Continuation of these bifurcations, as well as computing their normal forms, all in two control parameters.
- Computation of both stable and unstable one-dimensional invariant manifolds.
- Continuation of orbits with respect to a control parameter like α and the detection of tangencies on the curve of orbits.

There are a few more features, that we will not mention here, as they are not directly related to the thesis and the scripts we will write and/or edit here.

The main functionality we are interested in, is the continuation of invariant manifolds. MatContM does this in two distinctive steps, categorized as the **initializer** and the **continuer**. The initializer is also known as the **starter**, these terms will be used interchangeably.
Typically a user will use the MatContM GUI to define a map and provide and/or has the software provide its derivatives. Then the initializer produces the first point from which the continuation will start. This point can be point of the user's choice, but we can also compute a point on the invariant curve, if one is present. The initializer in this thesis will not produce a point however, but it will produce the start of the invariant curve instead. Then MatContM allows the user to see how this point changes if we iterate with f or how a fixed point or invariant curve evolves if we change f's control parameters. The latter is done by numerical continuation.

Chapter 2

Algorithms and Implementations

Before we can start the iterative continuation of our invariant curve, we need to specify a defining function F for it. Also, we need to starting point $x^{(0)}$ from which we will begin the continuation of the curve. This starting point is produced by the initializer. The main objective of this thesis is to develop a defining system and an initializer, and to include their numerical implementations in MatContM.

2.1 Defining system

Assume that our map (1.1) has a sufficiently smooth closed invariant curve C with an irrational rotation number ρ. Further assume that we can parametrize C by a smooth function $\theta \mapsto x(\theta)$ with $\theta \pmod{2\pi}$, such that

$$f(x(\theta),\alpha) = x(\theta + 2\pi\rho) \tag{2.1}$$

for all θ. According to Fourier theory, any sufficiently smooth periodic function can be written as a convergent trigonometric series. Next, we truncate the series to order N and now we need to find the coefficients $a_0, a_k, b_k \in \mathbb{R}^n$ $(1 \leq k \leq N)$ in the expression

$$x(\theta) = a_0 + \sum_{k=1}^{N} a_k \cos(k\theta) + b_k \sin(k\theta). \tag{2.2}$$

This is done by evaluating (2.1) at the equidistant points

$$\theta_j = \frac{2\pi j}{2N+1}, \qquad 0 \leq j \leq 2N.$$

This yields a system of $(2N+1)$ equations if ρ is fixed:

$$\begin{cases} f(x(\theta_0),\alpha) = x(\theta_0 + 2\pi\rho) \\ f(x(\theta_1),\alpha) = x(\theta_1 + 2\pi\rho) \\ \vdots \\ f(x(\theta_{2N}),\alpha) = x(\theta_{2N} + 2\pi\rho). \end{cases}$$

Each equation is in fact n-dimensional, so that we have here a system of $n(2N+1)$ scalar equations. Its solution is not unique since a translation $y(\theta) := x(\theta+\phi)$ with any ϕ also defines the same closed invariant curve. To make the solution unique one can set a component to be zero, e.g. $b_{1,1} = 0$ and remove it from the unknowns. Since ρ is fixed, one must add one more model parameter to the continuation variables to guarantee the solvability, as an invariant curve with rotation number ρ may not exist otherwise. Next one can turn this into a continuation problem by freeing one more model parameter.

Note that we have $(N+1)$ a_i-terms and N b_i-terms, with each n components, combined with two one-dimensional parameters. Thus, we have

$$M = (2N+1)n + 2$$

continuation variables. If we now define

$$X = (a_0, a_1, b_1, a_2, b_2, \ldots, a_N, b_N, \alpha_1, \alpha_2) \in \mathbb{R}^M$$

and set

$$\begin{cases} F_1(X) := f_1(x(\theta_0), \alpha) - x_1(\theta_0 + 2\pi\rho) \\ \vdots \\ F_n(X) := f_n(x(\theta_0), \alpha) - x_n(\theta_0 + 2\pi\rho) \\ F_{n+1}(X) = f_1(x(\theta_1), \alpha) - x_1(\theta_1 + 2\pi\rho) \\ \vdots \\ F_{M-2}(X) := f_n(x(\theta_{2N}), \alpha) - x_n(\theta_{2N} + 2\pi\rho) \\ F_{M-1}(X) := b_{1,1}, \end{cases} \tag{2.3}$$

where $x(\theta)$ is defined as in (2.2), we obtain the continuation problem

$$F(X) = 0, \quad F : \mathbb{R}^M \to \mathbb{R}^{M-1},$$

that will be needed to run the continuer. In this way we will approximate numerically a closed invariant curve with a fixed rotation number.

The defining system has been programmed in MATLAB, the code can be found in Appendix A.

2.2 Initializer

At the Neimark-sacker bifurcation, the critical center manifold W_0^c is tangent to the 2D eigenspace of A corresponding to the pair of eigenvalues $e^{\pm i\theta_0}$. This eigenspace can be parametrized by $z \in \mathbb{C}$ as

$$zq + \overline{zq} = 2\mathrm{Re}(zq),$$

where $q = q_R + iq_I \in \mathbb{C}^n$ is the normalized eigenvector associated to $e^{i\theta_0}$. This plane still approximates the parameter-dependent center manifold W_α^c for small $|\alpha| \neq 0$, when the map has a closed invariant curve. Thus, we can approximate this closed invariant curve in W_α^c using $z = \varepsilon e^{i\theta}$, i.e.

$$2\mathrm{Re}(zq) = 2\mathrm{Re}\left[\varepsilon(\cos\theta + i\sin\theta)(q_R + iq_I)\right] = 2\varepsilon(q_R\cos\theta - q_I\sin\theta),$$

where $\varepsilon > 0$ is a small user-defined parameter. Indeed, if we keep only the leading linear term in (1.9) and assume that $\beta = 0$, any closed curve $z = \varepsilon e^{i\theta}$ will be invariant with the rotation number

$$\rho_0 = \frac{\theta_0}{2\pi}.$$

The final initializer algorithm will look as follows. Take a fixed point x_0 at the Neimark-Sacker bifurcation parameter value α_0 and compute its complex eigenvector $q = q_R + iq_I$ corresponding to $e^{i\theta_0}$. Use as an approximation to the first closed invariant curve

$$x(\theta) = x_0 + 2\varepsilon(q_R\cos\theta - q_I\sin\theta)$$

with a small $\varepsilon > 0$. Furthermore, approximate the parameter value at with this curve will exist $\alpha = \alpha_0$ and use $\rho = \rho_0$ defined above as the constant rotation number in the defining system (2.3).

Since the defining system is formulated in terms of the Fourier coefficients, we have

$$a_0 = x_0, \quad a_1 = 2\varepsilon q_R, \quad b_1 = -2\varepsilon q_I.$$

Note that the eigenvector q should be scaled to satisfy $q_{I,1} = 0$ to assure that $b_{1,1} = 0$, which is always possible.

The way this has been implemented in MATLAB can be found in Appendix B.

2.3 Supporting scripts

Besides the two main scripts, we have also created a few scripts that support the two scripts described above. The FCMAP script, which is an edited script from Hil Meijer, evaluates the Fourier coefficients and returns the vector X as a vector of all components of a point x on the invariant curve evaluated at θ.

The testciv script is a function that accepts the starting fixed point and parameter values and a map, and then calculates where the bifurcation happens, calls the functions we implemented to initialize and continue the closed invariant curve and returns a big matrix with on each column the non-fixed Fourier coefficients of the closed invariant curve and under that the active parameter values corresponding to that curve. This is also an edited script from Hil Meijer. In both these scripts the edit includes generalizing the scripts to work on any given map.

The last script we used here is the PIC script, which is short for Plot Invariant Curve. We have written this script to accept any amount of columns from the matrix X outputted by the continuer, or equivalently, by the testciv script. It adds the zero on the fixed Fourier coefficient and draws all closed invariant curves defined by the columns in matrix X. The script accepts two arguments that define which dimensions should be plotted. The individual final curves are being drawn by standard MATLAB functions.

All of these scripts can be found in appendices C - E. In the scripts, some other scripts from MatContM are also called. These are unchanged and can be found in the current MatContM software.

Chapter 3

Examples

Of course we want to see that our methods work and verify that they in fact produce a series of invariant curves. As we want our methods to work for any map f that fits the criteria as described in Chapter 1, we will include two examples with different dimensions and parameters. This will ensure that the methods allow for a wide variety of maps to be analyzed. We will run the scripts we created using $N = 15$. Curves will be drawn using a different script that uses the standard MATLAB functionality to draw a continuous curve depending on one parameter θ.

3.1 A delayed logistic map

Our logistic map will keep track of the evolution of a population over time, where every step represents one year having past. As in a MatContM example on a logistic map [6], we will be looking at the recurrent relation

$$x_{k+1} = rx_k(1 - x_{k-1}) + \epsilon,$$

that depends on two parameters and where the next state depends on two previous states of the system. The r parameter represents the growth rate and the ϵ parameter represents the migration rate. To deal with this dependence on last year for calculating the value for next year, we introduce a variable

$$y_k = x_{k-1}$$

that keeps track of the previous year. Combining these, we can turn the equations into an iterative 2D map

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} rx(1-y)+\epsilon \\ x \end{pmatrix}. \tag{3.1}$$

Using the current functionality of MatContM, we can find fixed points of this map and we can find parameter values and a fixed point corresponding to these parameter values at which a Neimark-Sacker bifurcation occurs.

In figure 3.1 and figure 3.2 we have drawn the closed invariant curves as produced by our extension of MatContM.

Parameter values for figure 3.1

r	1.8182	1.8186	1.8203	1.8244	1.8312	1.8415	1.8567	1.8787	1.8925	1.9073
ε	0.1000	0.1007	0.1035	0.1098	0.1188	0.1300	0.1423	0.1544	0.1598	0.1647

Parameter values for figure 3.2

r	1.8182	1.8186	1.8203	1.8244	1.8312	1.8415	1.8567	1.8787	1.8925	1.9073	1.9167	1.9261	1.9384	1.9565	1.9871	2.0388	2.0539	2.0678	2.0751	2.0806	2.0853	2.0870	2.0831	2.0727	2.0574	2.0502	2.0796	2.1319
ε	0.1000	0.1007	0.1035	0.1098	0.1188	0.1300	0.1423	0.1544	0.1598	0.1647	0.1676	0.1702	0.1727	0.1742	0.1726	0.1644	0.1616	0.1591	0.1579	0.1571	0.1566	0.1566	0.1570	0.1573	0.1562	0.1510	0.1380	0.1201

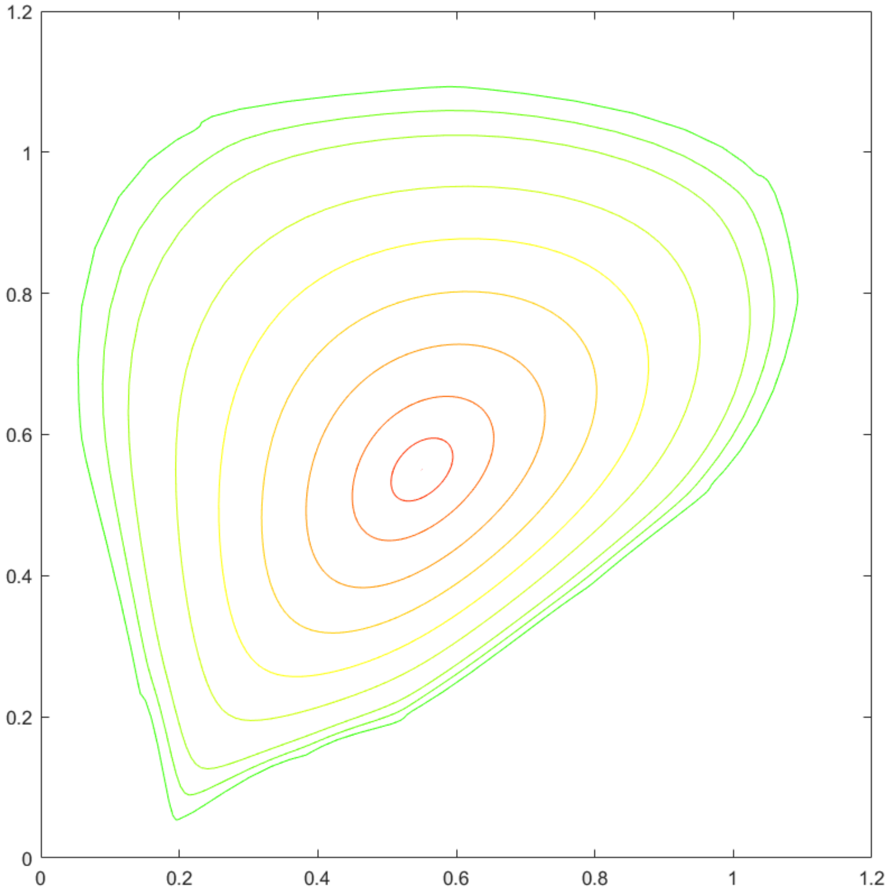

Figure 3.1: The evolution of the delayed logistic map's closed invariant curve bifurcating from the fixed point $x_0 = (0.55, 0.55)$ at $r = 1.8181, \varepsilon = 0.1$.

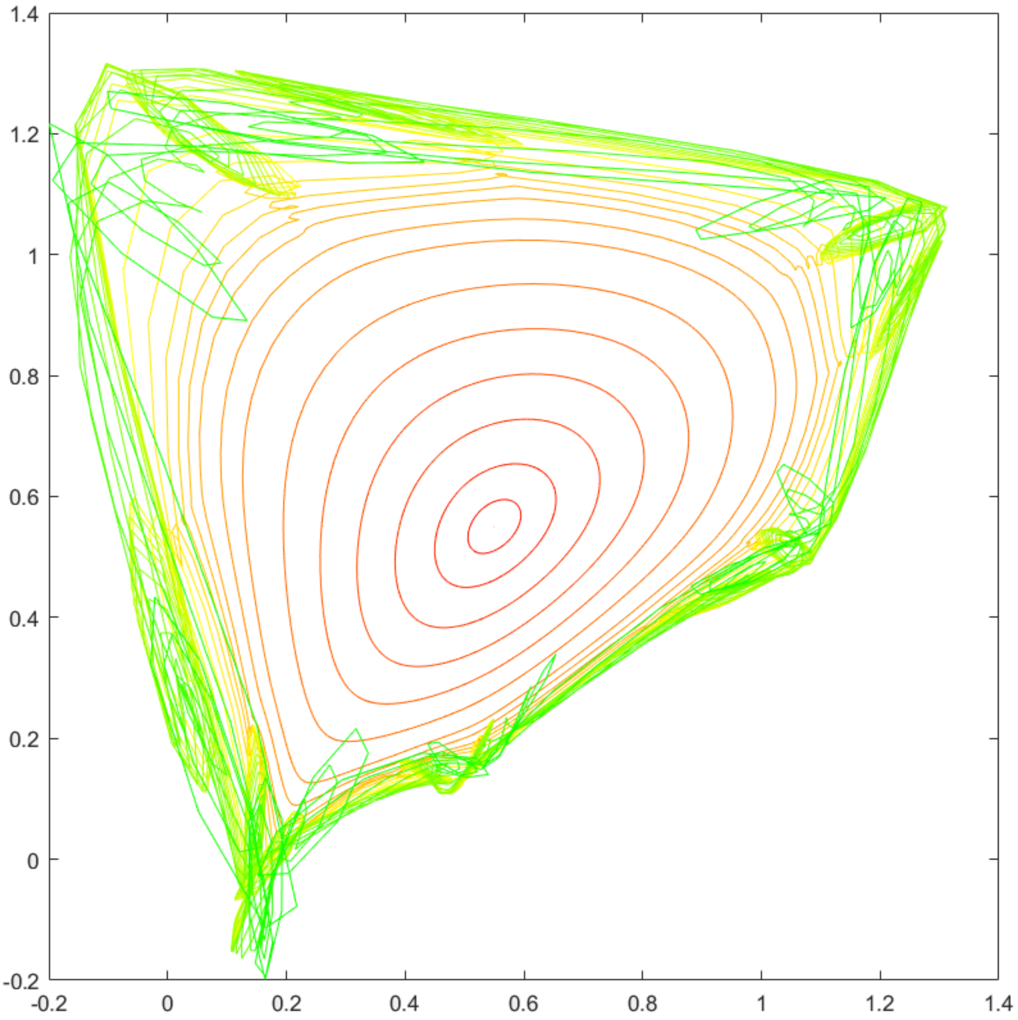

Figure 3.2: If we continue further, we see that the curve loses smoothness and starts to intersect itself.

3.2 An adaptive control map

The other example we will use is an adaptive control map introduced in [2]. This map is defined as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} y \\ bx + k + zy \\ z - \frac{ky}{c+y^2}(bx + k + zy - 1) \end{pmatrix}, \tag{3.2}$$

where b, k and c are parameters. This map is slightly more complicated and we use it here to illustrate that the MatContM software can also handle maps that have more than 2 dimensions and any amount of parameters. Keep in mind however that we do not vary all of these parameters when we continue its closed invariant curve. Again with the functionality of MatContM, we see that from the fixed point $x_0 = (1, 1, 0.3838)$ of (3.2) at parameter values $b = -0.5238, k = 1.14$ and $c = 0.1$, a closed invariant curve appears. This will then be the fixed point used by the initializer to find the first closed invariant curve.

We keep $c = 0.1$ constant here, as we only continue curves by varying two parameters. As we said, the interval in which the parameters still define a function with a closed invariant curve can vary quite a lot. In the adaptive control map, the twenty-second curve we calculated, or the outermost curve displayed in figure 3.3, has parameter values $b = -0.5122, k = 1.1965$ and $c = 0.1$. This gives a rough estimate of the parameter value ranges we deal with. In comparison, the last smooth curve we calculated for the delayed logistic map was found when the parameters r and ε of (3.1) had values 1.9073 and 0.1647 respectively.

In figures 3.3 and 3.4 the evolution of the invariant curve can be seen. The tables containing the parameter values that correspond to these curves can be found below as well.

Parameter values for figure 3.3

b	-0.5238	-0.5238	-0.5238	-0.5238	-0.5238	-0.5237	-0.5237	-0.5236	-0.5234
k	1.1400	1.1400	1.1400	1.1400	1.1400	1.1401	1.1401	1.1403	1.1405

b	-0.5230	-0.5224	-0.5216	-0.5206	-0.5195	-0.5181	-0.5165	-0.5148	-0.5130
k	1.1410	1.1418	1.1429	1.1444	1.1463	1.1488	1.1520	1.1560	1.1610

b	-0.5114	-0.5102	-0.5101	-0.5122
k	1.1671	1.1747	1.1841	1.1965

Parameter values for figure 3.4, continued from the values and curves from 3.3

b	-0.5146	-0.5179	-0.5201	-0.5223	-0.5247	-0.5274	-0.5303	-0.5337	-0.5355
k	1.2039	1.2117	1.2160	1.2198	1.2234	1.2265	1.2286	1.2287	1.2277

b	-0.5364	-0.5373	-0.5383	-0.5393	-0.5403	-0.5413	-0.5423	-0.5429	-0.5432
k	1.2271	1.2264	1.2256	1.2245	1.2226	1.2193	1.2136	1.2067	1.1998

b	-0.5445	-0.5464	-0.5491	-0.5530	-0.5575	-0.5617	-0.5641	-0.5639	-0.5604
k	1.1943	1.1933	1.1936	1.1953	1.1978	1.1998	1.2002	1.1986	1.1947

b	-0.5539
k	1.1893

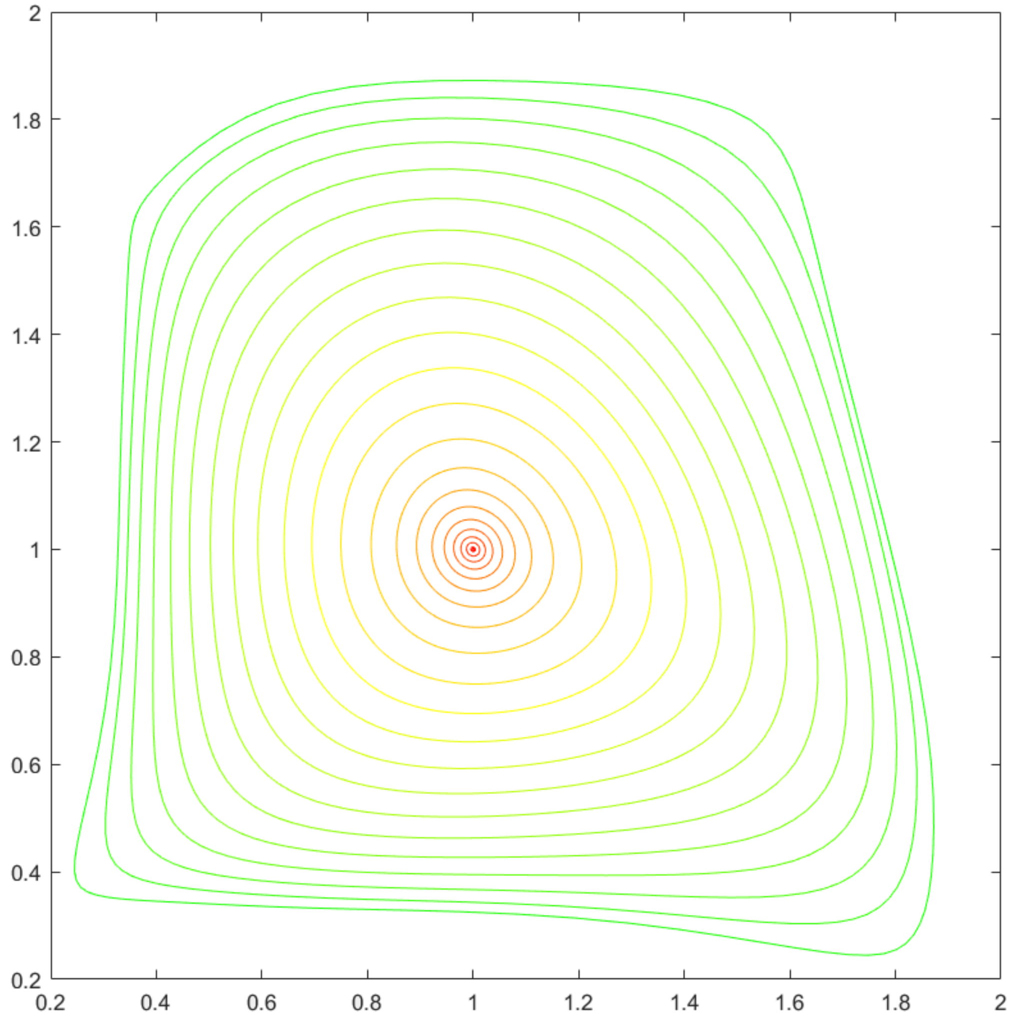

Figure 3.3: Here we see the first two dimensions of the closed invariant curve of the adaptive control map's closed invariant curve. Clearly the center is the point $x_0 = (1, 1, 0.3838)$ we calculated, restricted to the first two dimensions.

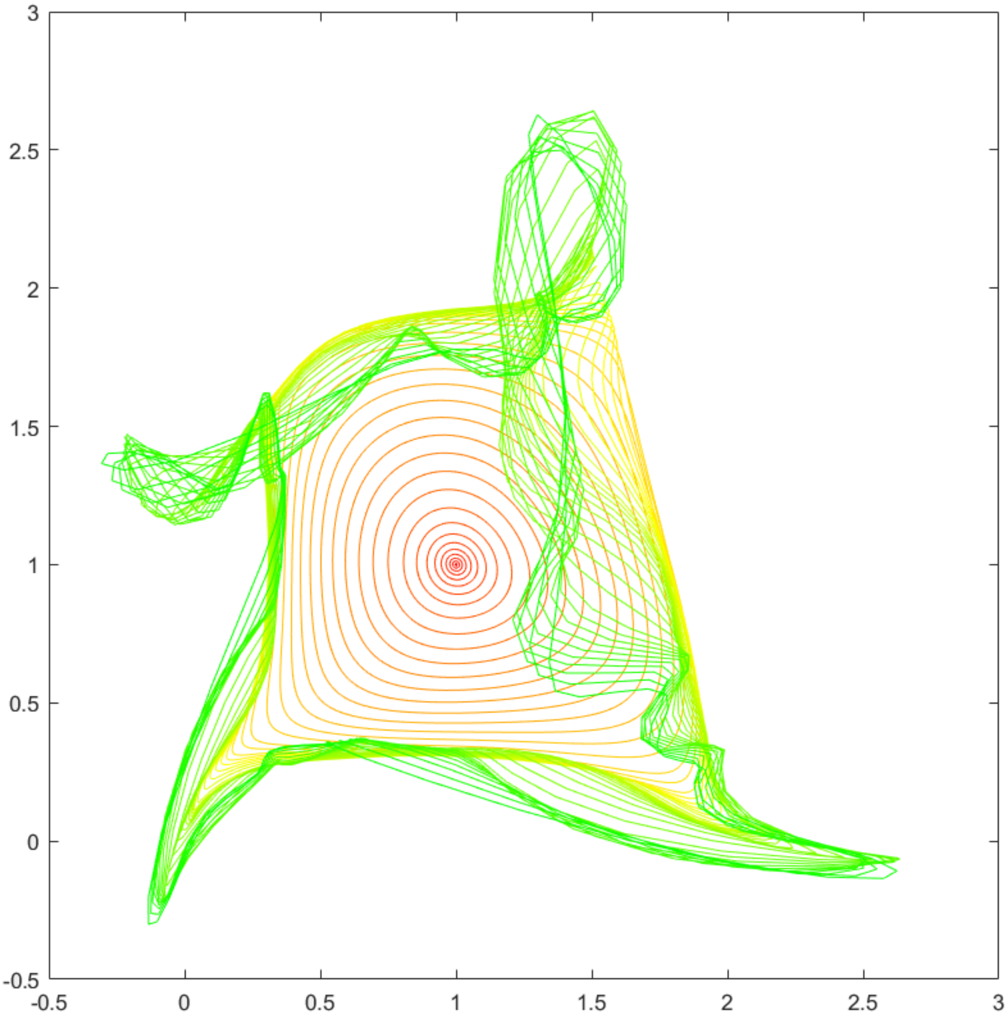

Figure 3.4: If we continue further, we see that the curve loses smoothness and starts to intersect itself.

Chapter 4

Conclusions

We can draw a few conclusions from the research that has been done while writing this thesis.

A smooth closed invariant curve can indeed be approximated by a convergent Fourier series of any length, depending on the desired precision. The defining system as described in Chapter 2 can be used by the current MatContM software to continue an existing closed invariant curve, both in forward and in backward direction.

Furthermore, the starting invariant curve as computed by our initializer does in fact produce a closed invariant curve that is very likely to be correct. We can conclude this from tests that we did with the examples we saw. We simply produce the invariant curve and then we let the continuer continue the closed invariant curve backwards with a small step size. The result is that we see that the closed invariant curve converges to the fixed point of which we know it is the corresponding Neimark-Sacker point. That means that, assuming that the defining system and the current software works correctly, the starting closed invariant curve has to be correct as well.

We also conclude that the parameter values for which this whole process produces useful results for us are limited. If we continue the curve too far, we see that it loses smoothness. The curve may also lose stability at some parameter values. Calculating at which exact values this happens, is something that we may want to add to the software in the future. Without that feature we can still estimate these values by analyzing the graphs however, as we have done shortly in the examples section.

Lastly, we know that the current functionality of MatContM has been added to a graphical user interface (GUI), to make it easier to use for users. As of now, the new methods have been implemented in such a way that they can be called via commands in the MATLAB command line. In the future, this will be added into the MatContM GUI as well.

In addition to this analysis, we can pick a point on the curve and use the current MatContM software to simulate the map from this point on wards, creating a forward orbit. This orbit should look the same as the computer closed invariant curve. In order to check this, we have taken the closed invariant curve and the curve generated by the forward orbit started on the curve at the same parameter values and put them in one figure, see Figure 4.1. We use this figure so also verify the functionality of the developed methods.

We also see that the non-smooth curves are not closed invariant curves, or curves that have any significant meaning, when we draw both these curves in the same figure. We see the result of this in Figure 4.2.

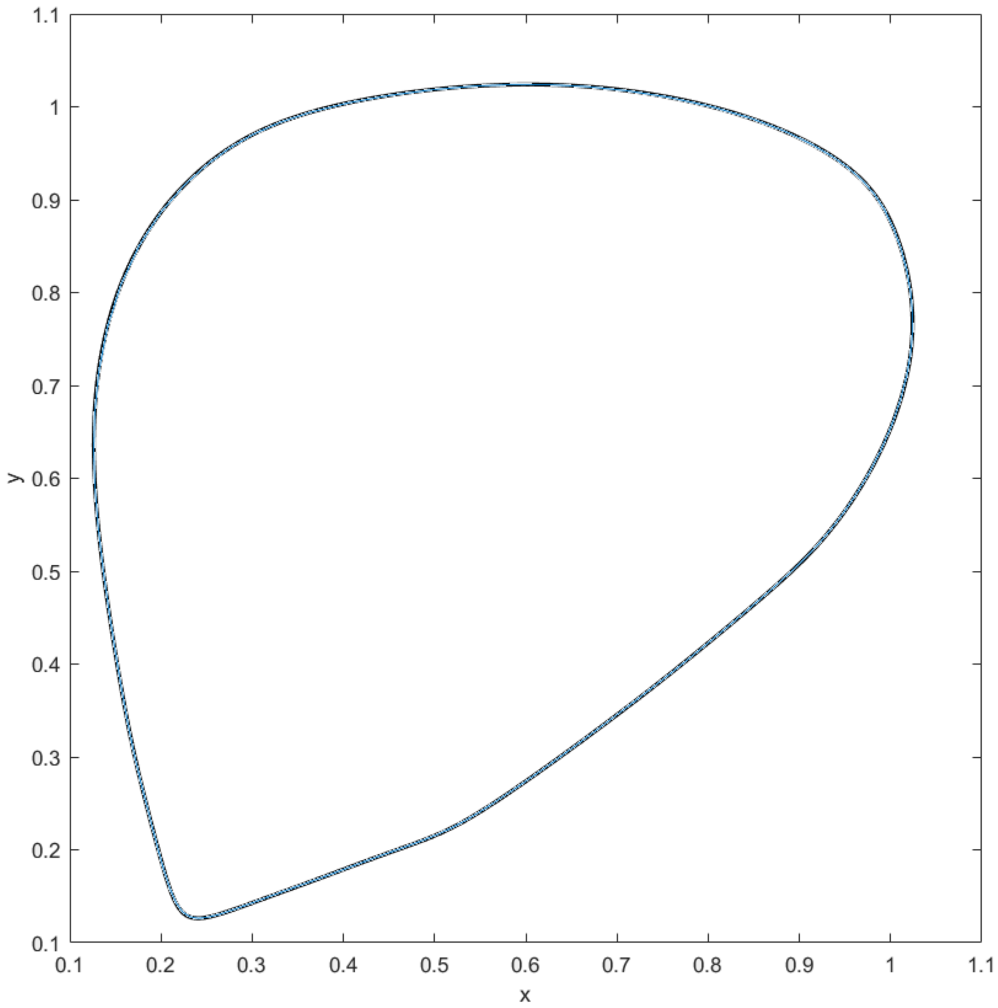

Figure 4.1: The blue line is the computed invariant curve, while the black line is the collection of points in the forward orbit of any point on the curve at $(r, \varepsilon) = (1.8787{,}0.1544)$. The graphs visually coincide, meaning that our predicted curve is very accurate.

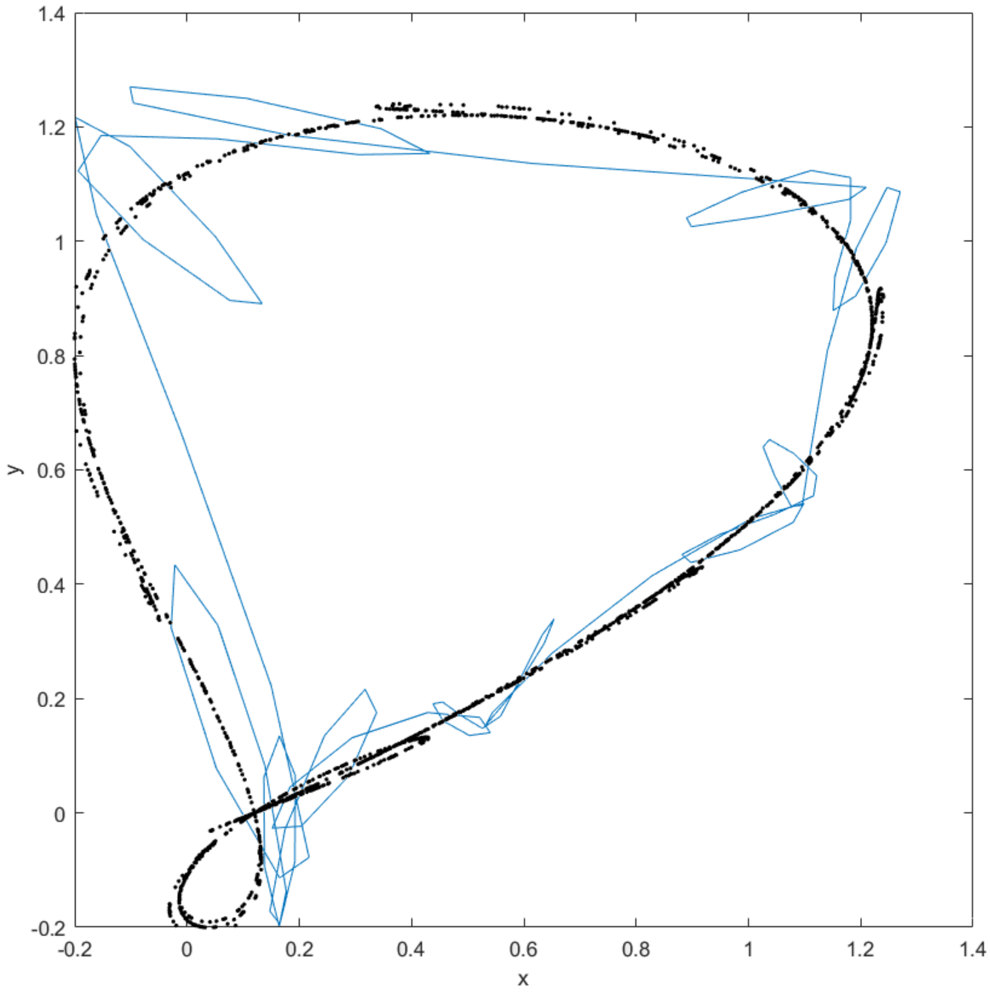

Figure 4.2: The blue line is the curve produced by the continuer when going to parameter values $r = 2.1319$ and $\varepsilon = 0.1201$. The black dots are the orbit produced by starting in a point on the curve and then simulating again with the MatContM. The loops in the computed closed curve approximately indicate where the actual chaotic fractal attractor of the map has more complicated structure.

Bibliography

[1] J. Bell. Denjoy's theorem on circle diffeomorphisms. pages 1–2, 4 2014. On-line lecture notes.

[2] R. Adomaitis C. Frouzakis and I. Kevrekidis. Resonance phenomena in an adaptively-controlled system. *International Journal of Bifurcation and Chaos*, 1(1):83–106, 1991.

[3] P. Hartman. A lemma in the theory of structural stability of differential equations. *Proceedings of the American Mathematical Society*, 11(4):610–620, 1960.

[4] Yu. A. Kuznetsov R. Khoshsiar Ghaziani N. Neirynck H.G.E. Meijer, W. Govaerts. Matcontm: A toolbox for continuation and bifurcation of cycles of maps: Command line use. 11 2017.

[5] Yu. A. Kuznetsov. *Elements of Applied Bifurcation Theory.* Springer-Verlag New York, 2004.

[6] Yu. A. Kuznetsov. Computer session V: Bifurcation analysis of maps with cl_matcontm. 1 2010.

[7] Yu. A. Kuznetsov and H. G. E. Meijer. *Numerical Bifurcation Analysis of Maps: From Theory to Software.* Cambridge University Press, 2019.

[8] E. Liz. A global picture of the gamma-ricker map: A flexible discrete-time model with factors of positive and negative density dependence. *Bulletin of Mathematical Biology*, 80:2, 12 2017.

[9] Yu. A. Kuznetsov O. Diekmann and W.-J. Beyn. Lecture notes on dynamical systems generated by ordinary differential equations and maps. pages 241–248, 12 2011.

[10] S. Neil Rasband. *Chaotic Dynamics of Nonlinear Systems.* Wiley-VCH, 1990.

Appendix A

The code for the defining system

```
function out = closedinvariantcurve
%
% Fixed Point of  Map curve definition file for a problem in mapfile
%

global cds fpmds
    out{1}  = @curve_func;
    out{2}  = @defaultprocessor;
    out{3}  = @options;
    out{4}  = @jacobian;
    out{5}  = @hessians;
    out{6}  = @testf;
    out{7}  = @userf;
    out{8}  = @process;
    out{9}  = @singmat;
    out{10} = @locate;
    out{11} = @init;
    out{12} = @done;
    out{13} = @adapt;
return
%-------------------------------------------------------
function dd = curve_func(arg)
global civds
FC = arg(1:end - length(civds.ap)); ps = arg(end-length(civds.ap)+1:end);
%FC = arg(1:end-2); ps = arg(end-1:end);

ps = num2cell(ps);
FC = [FC(1:2*civds.n+civds.zerocomponent); 0; FC(2*civds.n+1+civds.zerocomponent:end)];
    % Add a zero on spot b[1][zerocomp], which got removed before because it is fixed.
NN = civds.NN;
theta=2*pi*(0:2*NN)/(2*NN+1);
ind=[1:2*civds.n+civds.zerocomponent,2*civds.n+2+civds.zerocomponent:length(FC)]; %Keep
    zero component of sine fixed to zero, keep in mind n:m = {n, ..., m} inclusive
dd=zeros(civds.n,length(theta));

%Evaluate the map DD=F(x(t))-x(t+rho)
for ii=1:length(theta)
  dd(:,ii)=feval(civds.func,0,FCMAP(theta(ii),FC),ps{:},civds.pss{:})-FCMAP(theta(ii)+
      civds.rho,FC); %create all components of defining system
end

dd = reshape(dd, civds.n*(1+2*NN), 1); %put all values in one long array instead of a
    matrix
%---------------------------------------------------------------
function jac = jacobian(varargin)
global civds

```

```
NN=(length(FC)/civds.n-1)/2;
theta=2*pi*(0:2*NN)/(2*NN+1);
ind=[1:2*civds.n,2*civds.n+2:length(FC)]; %Keep first component of sine fixed to zero, keep in mind n:m = {n, ..., m} inclusive
eps=1e-4;
dd=zeros(civds.n,length(theta));

% Compute the Jacobian
% wrt Fourier coefficients
jac=zeros(length(FC));
parfor kk=1:length(FC)-1
  jj=ind(kk);
  d1=nan(size(dd));
  d2=nan(size(dd));
  F1=FC;F1(jj)=F1(jj)+eps;
  F2=FC;F2(jj)=F2(jj)-eps;
  for ii=1:length(theta)
    d1(:,ii)=feval(civds.func,0,FCMAP(theta(ii),F1),ps{:})-FCMAP(theta(ii)+civds.rho,F1);
    d2(:,ii)=feval(civds.func,0,FCMAP(theta(ii),F2),ps{:})-FCMAP(theta(ii)+civds.rho,F2);
  end
  jac(:,kk)=(reshape(d1,length(FC),1)-reshape(d2,length(FC),1))/(2*eps);
end
% %wrt System Parameter
% ps1=ps;ps2=ps;ps1{1}=ps1{1}+eps;ps2{1}=ps2{1}-eps;
% for ii=1:length(theta)
%   d1(:,ii)=feval(map,0,FCMAP(theta(ii),FC),ps1{:})-FCMAP(theta(ii)+civds.rho,FC);
%   d2(:,ii)=feval(map,0,FCMAP(theta(ii),FC),ps2{:})-FCMAP(theta(ii)+civds.rho,FC);
% end
% jac(:,end)=(reshape(d1,length(FC),1)-reshape(d2,length(FC),1))/(2*eps);
%
% end

   %---------------------------------------------------------------
function hess = hessians(varargin)

%---------------------------------------------------------------
function varargout = defaultprocessor(varargin)

global cds nsmds
%elseif strcmp(arg,'defaultprocessor')
  if nargin > 2
    s = varargin{3};
    varargout{3} = s;
  end
 % compute eigenvalues?
  if (cds.options.Multipliers==1)
      n=nsmds.Niterations;
      x0 = varargin{1}; [x,p] = rearr(x0); p = n2c(p);
      jac =nsmjac(x,p,n);
      varargout{2} = eig(jac);
  else
      varargout{2}= nan;
  end
  % all done succesfully
  varargout{1} = 0;
%-------------------------------------------------------------
function option = options
global cds nsmds
  option = contset;
  %----------------------------------------------------------------
function [out, failed] = testf(id, x, v)
%-----------------------------------------------------------------
function [out, failed] = userf(userinf, id, x, v)
%---------------------------------------------------------------------
function [failed,s] = process(id, x, v, s)

```

```
%------------------------------------------------------------
function [S,L] = singmat
global fpmds cds
% 0: testfunction must vanish
% 1: testfunction must not vanish
% everything else: ignore this testfunction

  S = [];

  L = [];

%--------------------------------------------------------
function [x,v] = locate(id, x1, v1, x2, v2)
%---------------------------------------------------------
function varargout = init(varargin)
%---------------------------------------------------------
function varargout = done

%----------------------------------------------------------
function [res,x,v] = adapt(x,v)
res = []; % no re-evaluations needed




%----------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ---------------------------------------------------------------

function [x,p] = rearr(x0)

% ---------------------------------------------------------------
function [x,v] = locateBP(id, x1, v1, x2, v2)

% ---------------------------------------------------------------

function [A, f] = locjac(x, b, p)

% ---------------------------------------------------------

function WorkspaceInit(x,v)

% ------------------------------------------------------

function WorkspaceDone

% -------------------------------------------------------


%SD:continues equilibrium of mapfile
```

Appendix B

The code for the initializer

```
function [x0,v0]= init_NSm_ICm(mapfile, x, p, NN,amp, ap,n)

global cds civds
zerocomponent = 0;   % if we would like to have our fixed component of b[1] in another place, we can put it here

cds.options.SymDerivative=0;
nphase=length(x);
civds.n = nphase; % The dimension of x, small n
civds.NumPars = length(p);
civds.mapfile = mapfile; % The map we want to evaluate
func_handles = feval(civds.mapfile);
civds.func = func_handles{2};
civds.NN = NN; %Number of Fourier cos+sin modes, excluding constant term
civds.p = p;
civds.ap = ap;
% initoption = 2; %as the initializer doesnt produce the first X yet, there are 2 hardcoded X0s from which we choose at the moment, depending on the system we chose
% rho = 0;

pss = civds.p(length(civds.ap)+1:end);
pss = num2cell(pss);
civds.pss = pss;

    A=cjac(civds.func,[],x,n2c(p)); % HGE: To be changed so that the Jacobian is called
    [V,D]=eig(A);D=diag(D);
    for i=1:nphase-1
      for j=i+1:nphase
        if (abs(D(i)*D(j)-1)<1e-3)
          if (imag(D(i))==0)
              debug('Neutral Saddle');return;
          elseif (imag(D(i))>0)
              idx=i;
          else
              idx=j;
          end
        end
      end
    end
    q=V(:,idx);
    q=exp(-1i*angle(q(1)))*q; %Ensure the first component of q is real by rotating q;
    x0=[x;amp*real(q);amp*imag(q);zeros(2*nphase*(NN-1),1)]; %padding by zeros as close to the NS-point only the first mode matters
    x0=[x0([1:2*nphase 2*nphase+2:end]); p(ap)];   %Omit the zero component from the continuation variable and add parameters.
    rho=angle(D(idx));

civds.zerocomponent = zerocomponent;
civds.rho = rho; % The initial curve radius
v0 = []; %initial tangent vectors
```

Appendix C

The code for FCMAP

```
function xx=FCMAP(theta,coef)
global civds
  xx=coef(1:civds.n); %a0 is our fixed point, found in the first n coefficients
  N=civds.NN;  %number of Fourier modes
  fc=reshape(coef(civds.n+1:end),civds.n,2*N); % Turn a1, b1, ..., aN, bN into Nxn matrix
  for ii=1:N
    xx=xx+fc(:,2*ii-1)*cos(ii*theta)+fc(:,2*ii)*sin(ii*theta);
  end

end
```

Appendix D

The code for testciv

```
% TEST CIV
function X4 = testciv(mapfile, firstX, firstP)

%% Preparing initial data from fixed point continuation and Neimark−Sacker
init;
% global opt cds fpmds;
opt = contset;
opt = contset(opt, 'Singularities', 1);
opt = contset(opt, 'Multipliers', 1);
opt = contset(opt, 'MaxNumPoints', 100);
x1 = firstX; %Test values for DLM and ACM [0.444444;0.444444] [1;1;0.38]
p1 = firstP; %Test values for DLM and ACM [1.8; 0.1] [−0.54;1.14;0.1]
[X10, V10] = init_FPm_FPm(mapfile, x1, p1,1,1);
[X1, V1, s1, h1, f1] = cont(@fixedpointmap, X10, V10, opt);

opt = contset(opt, 'MaxNumPoints', 50);
opt = contset(opt, 'IgnoreSingularity', [2 3 5]);
x3 = X1(1:end−1, s1(2).index);
p3 = p1; p3(1) = X1(end, s1(2).index);
[X30, V30] = init_NSm_NSm(mapfile, x3, p3, [1 2],15);
[X3, V3, s3, h3, f3] = cont(@neimarksackermap, X30, V30, opt);
addpath('ClosedInvariantCurve')

%% Extracting the NS−point
id=s1(2).index;x4=X1(1:end−1,id);p4=p1;p4(1)=X1(end,id);
N=15; %Number of Fourier modes used, only initializing the first
amp=0.02; %Initial amplitude
[X40,V40]=init_NSm_ICm(mapfile,x4,p4,N,amp,[1,2],1);
[X4,V4,s4,h4,f4]=cont(@closedinvariantcurve,X40,V40);
```

Appendix E

The code for PIC

```
% Plot invariant curves
function out = PIC(X, dim1, dim2)
global civds

% Extract the Fourier coefficients from X for each column
[n1,n2] = size(X); %n1 is the amount of rows, n2 the amount of columns

% For every column we plot the closed invariant curve
for col = 1:1:n2
    x = X(:,col:col);
    x = [x(1:2*civds.n+civds.zerocomponent); 0; x(2*civds.n+1+civds.zerocomponent:end-2)];
    theta = linspace(0,2*pi);
    V = FCMAP(theta,x);
    x1 = V(dim1:dim1,:);
    x2 = V(dim2:dim2,:);
    plot(x1,x2)
    hold on
end
hold off
end
```

Utrecht University

Faculty of Science
Department of Mathematics

Title of work:

Numerical continuation of closed invariant curves of maps and its implementation in the MATLAB software MatContM

Thesis type and date:

Bachelor thesis, June 2019

Supervision:

Prof. Dr. Yuri A. Kuznetsov
Dr. Hil G.E. Meijer

Student:

Name:	Luuk Sterke
E-mail:	luuk@sterke.net
Student number:	5740959