Investigating Dynamics by Multilevel Phase Space Discretization

Von der Fakultät
Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart
zur Erlangung der Würde eines Doktors
der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

Vorgelegt von

Danny Georg Fundinger

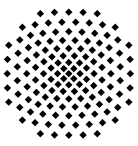
aus Pforzheim

Hauptberichter: *Prof. Dr. P. Levi*
Mitberichter: *Prof. Dr. G. S. Osipenko*
Tag der mündlichen Prüfung: *10.3.2006*

Abteilung Bildverstehen

Institut für Parallele
und Verteilte Systeme

Universität Stuttgart



IPVS

Institut für Parallele und Verteilte Syteme der Universität Stuttgart, 2006

Abstract

The subject of the thesis is the numerical investigation of dynamical systems. The aim is to provide approaches for the localization of several topological structures which are of vital importance for the global analysis of dynamical systems, namely, periodic orbits, the chain recurrent set, repellers, attractors and their domains of attraction as well as stable, unstable and connecting manifolds. The techniques introduced do not require any a priori knowledge about a system, and are also not restricted by the stability of the solution. Furthermore, they can generally be applied to a wide range of dynamical systems.

Two theoretical concepts are considered to be at the center of the research – symbolic analysis and the RIM method. The underlying basic approach for both of them is multilevel phase space discretization. This means that a part of the phase space, the area of investigation, is subdivided in a finite number of sets. Then, instead of each point of the phase space, only these sets are subject of further analysis. The main target of every method proposed is to find those sets which contain parts of the solution and subdivide them into smaller parts until a desired accuracy is reached.

In case of symbolic analysis, a directed graph is constructed which represents the structure of the state space for the investigated dynamical system. This graph is called the symbolic image of the focused system and can be seen as an approximation of the system flow. The theoretical background regarding the symbolic image graph as well as the constructive methods applied on it were already described in a series of works by G. Osipenko. In this work, strategies are introduced for a practical application. This requires the extension of the theoretical concepts and the development of appropriate algorithms and data structures. In practice, it turned out that these aspects are essential cornerstones for the usability of the discussed methods. Also some sophisticated tunings of the basic methods are proposed in order to extent the field of practical investigation.

Although symbolic analysis can be seen as the main stimulation of this work, the

investigation was not limited to it. Indeed, several shortcomings regarding the solution of some problems can be observed if the method is applied in practice. This led to the development of the RIM method. The core intention of the method is to solve the root finding problem. The standard approach toward this task is the application of an iteration scheme based on the Newton method. However, it has shown that such Newton schemes have several structural disadvantages which are especially crucial in the context of the fields of investigation which are relevant to this work. The RIM method proposes an alternative approach which does not require the application of any Newton-like method. Numerical case studies revealed that in several nontrivial scenarios the RIM method provides better results than both, symbolic analysis as well as Newton-based methods. Two applications of the RIM method for the investigation of dynamical systems are provided. One of them is the detection of periodic points. The other is the computation of stable manifolds.

The proposed methods contribute not only to the direct investigation and simulation of specific dynamical processes but also to the research in the field of dynamical system theory in general. This is due to the fact that progress in theory depends to a large extent on the observation and investigation of phenomenons. These phenomenons can often only be revealed, analyzed and verified by numerical experiments. The presented numerical case studies give some concrete examples for the application of the methods. Hereby, the dynamical models are taken from different fields of scientific research, like geography, biology, meteorology, or physics.

Acknowledgements

This work was carried out as a cooperative research project involving the Non-Linear Dynamics Group of the University of Stuttgart, Germany and the Laboratory of Mathematical Modeling at the St. Petersburg State Polytechnic University, Russia. This interdisciplinary environment was a highly stimulating and excellent workplace. I would like to thank all those people at both universities who contributed to this work with their commitment and invaluable advice.

It would have been impossible for me to accomplish this project without the generous support of Prof. Dr. Paul Levi and Prof. Dr. George Sergeevich Osipenko. The many fruitful and challenging discussions with Prof. Dr. Osipenko built the sound foundation of this work and encouraged me in my efforts to step deeper into the fascinating world of science.

The competent advice and ongoing support of Dr. Michael Schanz can also not be appreciated high enough. His personal confidence and scientific support were a strong backing during my work.

Although it has been a few years ago since Dr. Viktor Avrutin introduced me to the fascinating field of dynamical systems theory, his ideas and questions are still an ongoing source of inspiration and motivation which were truly invaluable for this work. The discussions with him deeply improved my knowledge. His expertise guided me through the grounds of dynamical systems theory.

The joint research work with Torsten Lindström from the University of Kalmar and Gunnar Söderbacka from the University of Lulea were a lively source of inspiration across the borders of scientific disciplines. Their interest in my work were a motivating force which encouraged me to proceed and extend my research.

Of no minor importance is the ongoing support of my family. Although sometimes it was probably not so easy for them to follow me on my way they never lost their confidence I am going the right path.

Last but not least, I have to thank the person who is the most important in my life – Yulia. Without you there would be a large gap in my life.

There are still many other people in Russia and Germany who contributed in one way or another to this work. I would like to thank all of those who encouraged and challenged me.

Contents

Chapter 1

Introduction

"The next best thing to knowing something is knowing where to find it." – **Samuel Johnson**

1.1 Motivation and Targets

The subject of this work is the numerical investigation of dynamical processes. Such dynamical, i.e. time-dependent, processes are a fundamental phenomenon which can be observed in plenty of different fields relevant to scientific research. Be it the motion of a pendulum in physics, the evolution of species in biology, or movements in the value of a currency in economics – all of these are time-dependent processes. These processes can be described by means of mathematics. The adequate tools to do so are dynamical systems, the mathematical models of time-dependent processes. Since the concept of dynamical systems was introduced in Newtonian mechanics, large efforts have been undertaken to acquire appropriate mathematical representations of natural and technical processes. This is not only the case for the classical fields of application, the natural sciences and engineering [AFH94], but also for others like medicine, sociology, meteorology and economics. Reason is that a time-dependent process can be better controlled, manipulated and predicted if it is modeled by a dynamical system. The analysis of the mathematical representation leads to a better understanding of the underlying process and also allows to study its behavior by numerical simulation. Hereby, dynamical systems theory is the branch of mathematics which focuses on the investigation of dynamical systems in general, and, hence, plays a key role for the understanding of every dynamical process.

Large progress has been made during the last 40 years in the field of dynamical systems theory. To a large extent, this is closely connected with the rapid

development during the same time regarding the numerical computation and investigation of dynamical systems. While in the 19th century the simulation of dynamical processes was still a time-consuming and costly work, it is in today's world by far cheaper and faster. The behavior of a dynamical system starting from an initial position can easily be predicted by computation of trajectories. This led not only to remarkable achievements for the practical application of mathematical modeling but also heavily influenced the analytical branch of dynamical systems theory.

In classical physics it was still assumed that, due to the deterministic nature of dynamical systems, the behavior of time-dependent processes is in principle computable and therefore also predictable – if only one has the computational power to do so. One of the most prominent supporters of this so-called *causal determinism* was Laplace, whose thoughts are expressed in the following quote:

> We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.

This intellect is often referred to as *Laplace's demon*. Ironically, it was the increase of computational power which revealed that a nonlinear dynamical system, though clearly deterministic in its future states, is not necessarily also predictable. In 1963, numerical experiments led to the discovery of chaotic, and, hence, unpredictable motion in the Lorenz attractor [Lor63]. This discovery resulted in a fundamental change of paradigms regarding the view on dynamical systems. The growing gap between non-predictable observations in real life processes and the expected predictability of dynamical systems had been closed. Notably, after the door had been opened, it has shown that a large richness of complicated dynamical behavior can be found even in very simple nonlinear systems, like the logistic map [May76]. The Mandelbrot and the Julia sets are some impressive classical examples of complicated, fractal structures [Man82] found in a simple dynamical system. The investigation of such nonlinear dynamics by purely analytical methods is, due to the complexity of the dynamics, in most cases very limited. Hence, modern theory of dynamical systems must rely to a large extent on the results of numerical exploration. Such investigations are of crucial importance in the field of dynamical systems theory and have heavily influenced

today's view on dynamical systems. The methods of numerical investigation are not only mere tools for the simulation of a specific process, but also essential cornerstones for the experimental development of the theory.

There are several approaches for the application of numerical investigation in the context of dynamical system theory. One of them is to reveal phenomena which might be subject of analytical research, as it was the case for the *Feigenbaum constant*. M. Feigenbaum first discovered this constant by numerical experiments. Then a mathematical proof of the fact was provided which applies to a wide class of dynamical systems [Fei83]. Other typical applications are the use of numerical simulation in order to compute measurements, e.g. the Lyapunov exponent [AFH94], or to apply theoretical results in complicated settings. For instance, the existence of stable and unstable manifolds has been proved theoretically but the detection of these manifolds can usually only be achieved by numerical methods.

Although the use of modern computers has greatly contributed to the investigation of dynamical systems, the development and application of numerical methods is still a challenging task. Reason is that the simulation of one trajectory, which is a considerably easy computational task, is usually not sufficient to capture the full dynamics. Instead, the global analysis of a dynamical system requires the simulation of virtually all trajectories. Obviously, this is an aim which can not be achieved by any finite numerical computation. Appropriate methods are needed which approximate a system's global dynamics at least partly. Hereby, one speaks mainly of two general approaches, the *direct* and *indirect* methods, see e.g. [Bey90].

The direct methods focus on the development of numerical techniques that find particular dynamical structures, like periodic orbits, invariant manifolds etc. Knowledge about the topology of an object is used to develop the numerical method. Often, the aim is to construct a problem of finding zeros, and then solve this problem with the Newton method. Direct approaches can provide very exact solutions of a problem. However, they are limited to the detection of particular structures. Often they are also only useful for local analysis, i.e. they require a priori knowledge about the position of an object and are not capable to approximate objects whose existence and position is yet unknown.

The indirect methods are closely related to the simulation of trajectories. The aim is to develop numerical methods which exhibit the same dynamics as the original system. Usually, this means that the system's dynamics are approximated by the computation of parts of a limited number of trajectories. These methods are

always applicable and the computations are usually quite cheap. Furthermore, no a priori knowledge about a system is required for the application. However, only those structures of a system can be captured which possess a stable long-term behavior. Other structures, which might also be important for the analysis of the system's dynamics, can not be revealed.

Although both approaches are essential tools for the numerical investigation, they are not yet sufficient for the global analysis of dynamical systems. What is needed are numerical methods which do not require any a priori knowledge about the position of a structure and are also capable to reveal those structures which do not possess stable long-term behavior. A successful approach to achieve this aim is given by methods which are based on the discretization of the phase space. Such methods can not be clearly categorized as direct or indirect. Rather more, they are a combination of both approaches.

This work is focused on such methods of phase space discretization. It is our aim to provide new approaches toward the computation of several topological structures which are of crucial importance for the global analysis of dynamical systems, like, for instance, periodic orbits, attractors and (un)stable manifolds. For some of these investigations other approaches already exist. Whenever necessary, we will later point out the reasons why we nevertheless decided to propose new ones. In most of these cases we do not consider our methods as superior to existing ones but rather more as alternatives which might be advantageous in some situations. Generally, the methods we propose should be capable of finding structures without any a priori knowledge and without any restrictions concerning the stability of the solutions. Furthermore, they should be applicable to a wide class of dynamical systems.

The introduced techniques are motivated by and partly based on symbolic analysis, a conceptual framework for the global analysis of dynamical systems. However, not all computational problems could be solved satisfiable within this framework. Hence, alternative approaches are required. This led to the development of the RIM method, another framework which is related to symbolic analysis but advantageous for the solution of several problems. A more detailed discussion about symbolic analysis, RIM and related concepts follows in the next section.

Several steps of development are required in order get the appropriate tools for the computation. First of all, the mathematical background must be set. Each method must be described mathematically, and it is necessary to prove that it solves the intended problem correctly. Then, the formulation of an appropriate

algorithm must be given. Again, correctness of this algorithm should be proved. Furthermore, for a practical application one has to consider that a numerical technique should not only be correct but also efficient. Hence, is is necessary to analyze the performance of a method and also consider details about the implementation, like specific data structures. As a last step, numerical case studies of a concrete implementation have to be performed. Obviously, the entirety of these steps requires not only aspects linked with the mathematical field, but also with the field of computer science.

It was our intention to give an overview about all these steps of development. For some techniques, the mathematical background was already given, e.g. in case of symbolic analysis, and our focus was put on the development of a correct and efficient implementation. For other techniques, like those based on the RIM method, no preliminary knowledge exists, and the mathematical foundation is also studied. As a main result of our investigation, a software was developed which allows the practical application of the numerical techniques presented here. This software is part of the larger non-commercial package AnT [ALS+03] and available for download, see [ant05].

1.2 Fundamental Conceptual Frameworks

The basic technique for all our numerical methods is *multilevel phase space discretization*. Two basic concepts are considered to be its essentials. These are the discretization of the phase space and the multilevel subdivision of the discretized areas. Discretization of the phase space means that a part of the phase space, the so-called *area of investigation*, is subdivided in a finite number of sets. Then, instead of each point of the phase space, only these sets are subject of analysis. Note that such a discretization is a natural assumption for numerical computations. In fact, every numerical computation requires a discretization given at least by the finite number of computed digits of its processed numbers. However, the discretization assumed in this context is usually much coarser. In combination with the second concept mentioned, the multilevel subdivision, a rough initial discretization is assumed which gets finer during several steps of computation. More precisely, after the phase space has been discretized, several elements will be selected. Each of these selected elements is subdivided into smaller parts. By doing so, a finer discretization of selected areas of the phase space can be achieved. This process is repeated for several times, until a desired precision of the discretization has been reached. This technique is called multilevel phase space discretization, or simply MPSD. It can be considered as the core computational scheme which is required for all our numerical methods. Actually, the principle computations of all

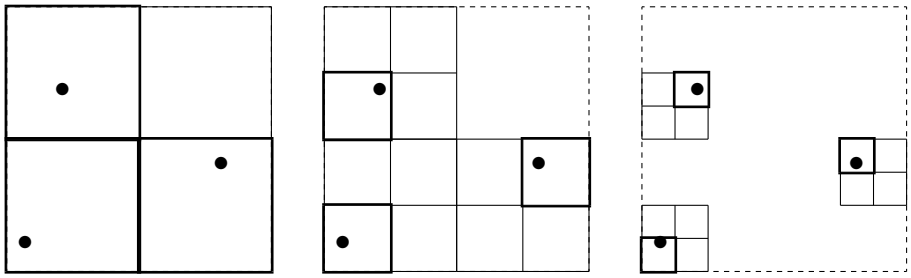


Figure 1.1: Example of multilevel phase space discretization.

our methods are embedded into MPSD. Each method we propose is mainly concerned about the decision if a part of the phase space gets selected for subdivision or not. The selected parts are those which contain a solution of the problem that the specific method is trying to solve. If, for instance, the problem to be solved is the localization of periodic points then the task of our numerical investigation is to decide for a discretized area of the phase space if it contains at least one of these periodic points. If so, the area is selected for subdivision, and the numerical investigation must be repeated for every sub-part of the area. Obviously, the application of MPSD always only produces an *outer covering* of the real solution. The precision of this outer covering depends on the discretization of the phase space after the final subdivision step. Furthermore, the analysis is limited to the area of investigation.

Example 1.1. *An example of the method is illustrated in Fig. 1.1. The solution is marked by* 3 *points. The phase space is initially divided in* 4 *boxes. The boxes which contain a solution are selected for further subdivision. Other areas are not investigated. After* 3 *steps of subdivision the selected boxes give a rather fine outer covering of the solutions.*

The techniques of phase space discretization and subdivision are well-known and widely appreciated in numerics. For instance, the methods of *interval analysis* perform a phase space discretization which is often combined with a subdivision scheme, see Alefeld and Herzberger [AH83], Hansen [Han92], Kearfott [KN90, Kea96] and references therein. Another example are some related methods of *global optimization*, see e.g. Horst and Tuy [HT90], Pintér [Pin96], and Huyer and Neumaier [HN99]. Close to our fields of application are also the tools for discretization of dynamical systems which were applied by F.S. Hunt [Hun98] and Diamond *et.al.* [DKP95].

MPSD is considered to be only the basic conceptual layer for the techniques proposed in this work. Built upon are the theoretical concepts of *symbolic analysis*. Basically, symbolic analysis provides a unified framework for the acquisition of information about the flow of a dynamical system without any restrictions concerning the stability of specific invariant sets. The mathematical theory was presented in a series of works by G. S. Osipenko [Osi83, Osi93, Osi94, Osi04]. It can be considered close to *Cell-to-Cell mappings* [Hsu80, Hsu87] and is related with *symbolic dynamics* [Ale76, Bow82, LM95, Wal91]. The main idea of symbolic analysis is the construction of a directed graph which represents the structure of the state space for the investigated dynamical system. This graph is called the *symbolic image* of the focused system and can be seen as an approximation of the system flow. It is connected with MPSD by the fact that each vertex of the graph is an area of the discretized phase space, and a successive refinement is achieved by multilevel subdivision. From the computational point of view, the usage of such a graph bears the big advantage that, once it is constructed, all investigations are matters of graph analysis. For instance, each strongly connected part represents a component of the chain recurrent set of the flow. More sophisticated computational analysis of the symbolic image graph allows, among others, the localization of periodic orbits [Osi93], invariant sets [KMO03], attractors and their basins [OC99, Osi99] as well as the computation of the *Morse Spectrum* [Osi97, Osi00, ORAP04] or verification of hyperbolicity. A comprehensive overview can be found in [Osi04].

Several other approaches exist which use concepts similar to the construction of the symbolic image graph. In Eidenschink [Eid95] and Mischaikow [Mis02] a symbolic image-like graph, called there a *multivalued mapping*, is constructed in order to compute isolated blocks in the context of the *Conley Index Theory*, see also [Con78]. The *set-oriented* methods of Dellnitz, Hohmann and Junge [DH96, DH97, DJ98, DJ02] use a scheme similar to symbolic analysis and apply a subdivision technique which is also used slightly modified in our implementation. Hruska [Hru02, Hru05] makes a *box chain construction* to get a directed graph with the aim to compute an *expanding*, or *hyperbolic*, metric for dynamical systems.

The theoretical background regarding the symbolic image graph as well as the constructive methods applied on it were already described in detail by Osipenko. But although in [Osi93] first numerical calculations were presented, the algorithmic basics, which are needed for the construction of the graph were not reported until now. Therefore, in this work we introduce algorithms and data structures which are important for an efficient implementation. In practice, it turned out that these aspects are essential cornerstones for the usability of the discussed

methods. However, this is not yet considered the main task of this work. Instead, the focus is put on some of the investigation methods on the graph. For their practical application, the theoretical concepts must be extended, and algorithms have to be developed. Also, we propose some sophisticated tunings of the basic methods in order to extent the field of practical investigation.

Although symbolic analysis can be seen as the main stimulation of this work, our investigation was not limited to it. Indeed, by its practical application we observed several shortcomings regarding the solution of some problems. It proved to be difficult to overcome these problems within the given framework. This led to the development of the RIM method, which fills the second half of this work. This method, though also based on MPSD, does not fit into the framework of symbolic analysis. Hence, not only its implementation but also the underlying theoretical concepts are introduced here.

The core intention of the RIM method is to solve the root finding problem, or, in other words, to find all the zeros of an equation. It was already mentioned in the context of direct methods that several problems regarding the global analysis of dynamical systems can be formulated as such root finding problems. The standard approach to solve them is to apply an iteration scheme based on the Newton method. However, it has shown that such Newton schemes have several structural disadvantages which are especially crucial in the context of our fields of investigation, see also Chapter 6 for a more detailed discussion. For this reason, the RIM method proposes an alternative approach which does not require the application of any Newton-like method. Numerical case studies revealed that in several nontrivial scenarios the RIM method provides better results than both, symbolic analysis and Newton-based direct methods.

1.3 Outline

This work is structured as follows:

Chapter 2 gives an overview about the fields of investigation. We define which kind of information about a dynamical system we like to acquire by our numerical methods, and what are the main problems of a numerical computation.

Chapter 3 describes the implementation of the basic concepts of symbolic analysis. We give details about data structures and algorithms. Furthermore, the characteristics of this implementation are analyzed. Some numerical case studies are given to demonstrate the practical application.

Chapter 4 gives a summary about several extensions and tunings for our proposed implementation. Our focus is put on techniques which improve the practical application of the investigation methods. Hereby, we respond to the results of our numerical case studies, and introduce techniques which are not considered in the theoretical context.

Chapter 5 focuses on advanced investigation methods based on symbolic analysis. For an application of these methods, it is necessary to extend the existing theoretical concepts. Furthermore, algorithms are introduced and analyzed. Numerical case studies demonstrate the application of the investigation methods.

Chapter 6 introduces the RIM method. The underlying theoretical concepts as well as an implementation of them are subject of discussion. Also, comparisons with related concepts are given. Hereby, a strong focus is put on the characteristics of a practical application.

Chapter 7 describes how the RIM method can be applied for the investigation of dynamical systems. Several changes and extensions of the basic framework are necessary to do so. Numerical case studies demonstrate the application of the methods.

Chapter 8 provides a summary of the achievements of our studies. These achievements are also compared with those of others. The chapter ends with an outlook about possible further research.

Note that ideas of Chapters 3, 5 and 5 were published in [AFL+06, Fun05, FO03].

Chapter 2

Fields of Investigation

In this chapter we give an introduction to the fields of investigation. More precisely, we establish the terminology which we will use throughout this work and provide mathematical definitions for the structures we want to compute by our numerical methods. These structures are usually subsets of a C^∞-smooth manifold which fulfill certain conditions in the context of the dynamical system in focus. The motivation for their computation is also considered. It is our intention to clarify why a field of investigation is of interest, and what kind of information about the global structure of the dynamical system can be acquired from its numerical analysis. Furthermore, we discuss some basic considerations about the computation of these entities and give an overview about existing approaches of numerical investigation.

The chapter starts with a definition of the classes of dynamical systems for which our methods are applicable. Then we introduce the different fields of investigations, namely periodic orbits, the chain recurrent set, attractors, repellers, the basin of attraction, connecting manifolds, stable and unstable manifolds as well as filtrations.

2.1 Dynamical Systems

Let $\mathcal{M}$ be a C^∞-smooth manifold which is a compact in $\mathbb{R}^d$, and $\mathbb{T}$ be a time space with $\mathbb{T} \in \{\mathbb{Z}, \mathbb{Z}^+, \mathbb{R}, \mathbb{R}^+\}$. A dynamical system is a continuous mapping $\phi(\mathbf{x},t)$ with $\mathbf{x} \in \mathcal{M}$ and $t,s \in \mathbb{T}$, so that $\phi : \mathcal{M} \times \mathbb{T} \mapsto \mathcal{M}$,

$$\phi(\mathbf{x},0) = \mathbf{x}, \tag{2.1}$$

$$\phi(\phi(\mathbf{x},t),s) = \phi(\mathbf{x},t+s). \tag{2.2}$$

In case $t,s \in \mathbb{T} \subseteq \mathbb{Z}$ we talk about a dynamical system discrete in time, and in case $t,s \in \mathbb{T} \subseteq \mathbb{R}$ we have a dynamical system continuous in time. In this work, both types of systems are of relevance though we mainly focus on dynamical systems discrete in time.

Let $\mathbf{f} : \mathcal{M} \mapsto \mathcal{M}$ be a continuous mapping and let $\mathbb{Z}^T = \mathbb{T} \subseteq \mathbb{Z}$ be a discrete time space. Then $\mathbf{f}$ generates a dynamical system discrete in time of the form $\phi(\mathbf{x},k) = \mathbf{f}^{[k]}(\mathbf{x})$, $k \in \mathbb{Z}^T$. The cascade

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k), \mathbf{x}_k \in \mathcal{M} \subset \mathbb{R}^d$$

describes its dynamics. The reduction of the time space to $\mathbb{Z}^T = \mathbb{T} \subseteq \mathbb{Z}$ ensures the existence of solutions for all $\mathbf{x}_0 \in \mathcal{M}$ and $k \to \pm\infty$. If $\mathbb{T}$ equals $\mathbb{Z}$ the dynamical system is called *invertible*. In such a system an initial state $\mathbf{x}_0$ uniquely defines the future states of the system as well as its past behavior. If a system is *noninvertible* then $\mathbf{x}_0$ only defines the future states uniquely and its past behavior is not unique, i.e there are typically no unique solutions for $k < 0$. In such a case the time space $\mathbb{T}$ equals $\mathbb{Z}^+$. The theoretical works of Osipenko, which are the basis for our numerical methods, are restricted to dynamical systems generated by homeomorphisms. By definition, these systems are invertible, i.e. the inverse $\mathbf{f}^{[-1]}$ of the system function $\mathbf{f}$ exists. However, the numerical methods we present in this work are generally also applicable for noninvertible systems. Hence, we also consider this class of dynamical systems.

Let us next consider a dynamical system which is continuous in time and given by an ordinary differential equation (ODE). In order to describe this system, we consider a shift operator along trajectories. Let $\dot{\mathbf{x}} = \mathbf{F}(t,\mathbf{x})$ be a system of ordinary differential equations, whereby $\mathbf{x} \in \mathcal{M}$ and $\mathbf{F}(t,\mathbf{x})$ is a C^1 vector field periodic in $t \in \mathbb{T} \subseteq \mathbb{R}$ with period ω. Denote the solution of such a system by $\phi(t,t_0,\mathbf{x}_0)$ with an initial condition $\phi(t_0,t_0,\mathbf{x}_0) = \mathbf{x}_0$. For the investigation of the global evolution of the system, it is usually sufficient to examine the Poincaré map $\mathbf{f}(\mathbf{x}) = \phi(\omega,0,\mathbf{x})$ which is the ω-shift operator along the trajectories of the system. In case the system is autonomous, i.e. the vector field does not depend on t, we fix an arbitrary $\omega \neq 0$ and consider a shift operator of the form $\mathbf{f}(\mathbf{x}) = \phi(\omega,\mathbf{x})$, whereby $\phi(t,\mathbf{x})$ is the solution of the autonomous system, $\phi(0,\mathbf{x}) = \mathbf{x}$. Note that the shift operator is a continuous mapping, or, more precisely, a diffeomorphism. Hence, it transforms the original system continuous in time into a dynamical system discrete in time. Consequently, in the following we can only consider discrete dynamical systems. If the underlying system is continuous in time and given by an ODE, we construct a shift operator in order to transform it into a discrete form. In case such a shift operator can not be constructed, the system

is not suitable for our investigations. In Sec. 4.1.1 the construction and practical application of a general shift operator for ODEs is discussed in more detail.

So in the following we consider systems discrete in time which are generated by a mapping $\mathbf{f} : \mathcal{M} \mapsto \mathcal{M}$. Such a mapping $\mathbf{f}$ is continuous but not necessarily invertible. This implies that we can also consider systems which are continuous in time if they can be transformed into discrete ones by a shift operator. In that case, the mapping $\mathbf{f}$ is considered to be the shift operator. Note that such a shift operator $\mathbf{f}$ is a diffeomorphism and, hence, also invertible.

We proceed by introducing some fundamental concepts for dynamical systems discrete in time. The trajectory (or orbit) of an initial point $\mathbf{x}_0$ is a sequence

$$T(\mathbf{x}_0) = \left\{ \mathbf{x}_k = \mathbf{f}^{[k]}(\mathbf{x}_0), k \in \mathbb{Z}^T \right\}.$$

In case of an invertible system, we say that

$$\begin{aligned} T^+(\mathbf{x}_0) &= \left\{ \mathbf{f}^{[k]}(\mathbf{x}_0), k \in \mathbb{Z}^+ \right\}, \\ T^-(\mathbf{x}_0) &= \left\{ \mathbf{f}^{[k]}(\mathbf{x}_0), k \in \mathbb{Z}^- \right\} \end{aligned}$$

are the positive and negative semi-trajectory. The computation of trajectories is the most basic and probably most widely used approach for the numerical investigation of dynamical systems. A lot of information about a system can be gathered by calculation and analysis of trajectories starting from several initial points. It is clear that only limited parts of a whole trajectory can be computed numerically. We say that we apply a *forward iteration* if a finite part of a positive semi-trajectory is computed, and that we apply a *backward iteration* if a finite part of a negative semi-trajectory is approximated. Note that a backward iteration can only be applied if the inverse $\mathbf{f}^{[-1]}$ exists and if it can be computed either analytically or numerically.

Besides the trajectories, another fundamental concept is the *invariant set*. A set $Q \subset \mathcal{M}$ is said to be invariant if $\mathbf{f}(Q) = Q$. This means that Q is a union of orbits, and that orbits starting in Q reside in Q. Some properties of invariant sets are that a closure, a union and an intersection of invariant sets are invariant as well. It is clear that in case $\mathbf{f}$ is a homeomorphism, an invariant set is also invariant for the inverse mapping $\mathbf{f}^{[-1]}$.

The stability of an invariant set is described according to Lyapunov. Let us first denote a ρ-norm distance on $\mathcal{M}$ by $\rho(\mathbf{x},\mathbf{y})$, e.g. $\rho(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|$. A distance

between a point $\mathbf{x}$ and a set A is defined as

$$\rho(\mathbf{x},A) = \inf\left(\rho(\mathbf{x},\mathbf{y}) \mid \mathbf{y} \in A\right).$$

An ε-neighborhood of A is denoted by $V(\varepsilon,A) = \{\mathbf{x} \mid \rho(\mathbf{x},A) < \varepsilon\}$, assuming $\varepsilon > 0$.

Definition 2.1. *An invariant set Q is called*

1. Lyapunov stable*, if for every $\varepsilon > 0$ there exists $\delta > 0$ so that if $\mathbf{x} \in V(\delta,Q)$ then the positive semi-trajectory $T^+(\mathbf{x}) \subset V(\varepsilon,Q)$,*
2. attracting*, if there is a neighborhood V such that $\lim_{k\to\infty} \rho(\mathbf{f}^{[k]}(\mathbf{x}),Q) = 0$ for all $\mathbf{x} \in V$,*
3. asymptotically stable *if it is Lyapunov stable and attracting.*

An invariant set can be classified as being either asymptotically stable (or simply stable) or unstable.

Stability is an important property of an invariant set. In the context of numerical computations, stable invariant sets are usually much easier to compute than unstable ones. Reason for this is that a stable set can be localized by starting a trajectory in its attracting neighborhood. By definition, this trajectory will approach the invariant set for $k \to \infty$. In practice, such a forward iteration usually comes reasonably close to the invariant set after a limited number of n steps. Then every further iteration of the trajectory T^+, i. e. all $x_k \in T^+$ for $k > n$, approximates parts of the invariant set. The quality of this approximation depends only on the number of forward iterations and the type of invariant set. This is not the case for unstable sets. Trajectories started in the neighborhood of an unstable invariant set do not approach this set. Hence, an approximation is only possible if the initial point of the trajectory already belongs to the set. This is usually difficult to achieve. Furthermore, even if the initial point belongs to the invariant set, an approximation is only possible if the numerical error of the forward iteration is so small that no point of it lies in the neighborhood of the invariant set because if a trajectory belongs to the neighborhood of an unstable invariant set, it is not attracted by this set and usually drifts away from it.

Remark 2.1. *One might think now that, if stable invariant sets can be approximated by forward iteration, then it might be possible to approximate unstable invariant sets by starting a backward iteration in its neighborhood. Although in some cases, like for repellers, this is true, in general, such a conclusion is not valid. Some kinds of unstable invariant sets can neither be approached by forward nor by backward iteration starting in its neighborhood.*

One of the main advantages of the numerical methods we propose in this work is that they are capable of approximating several kinds of unstable invariant sets. This can not be achieved by numerical methods based on the application of forward iterations.

2.2 Periodic Points

The detection of periodic orbits is an essential task for the analysis of nonlinear dynamical systems discrete in time. Distinctive features which are crucial for the understanding of a system's dynamics, like attractors, repellers or saddles, are in many cases represented by periodic orbits. Furthermore, unstable periodic orbits are considered to be fundamental building blocks of chaotic attractors [ACE$^+$87, Cvi92, GOY88], and thus also of a special interest for the study of chaotic dynamics .

A point $\mathbf{x}_0$ is called p-periodic with period $p \in \mathbb{N}^+$ if $\mathbf{f}^{[p]}(\mathbf{x}_0) = \mathbf{x}_0$. The smallest positive period p is called the *least period* . If the least period is $p = 1$ then $\mathbf{x}_0$ is called a *fixed point*. The trajectory (or orbit) of a periodic point $\mathbf{x}_0$ with the least period p consists of p different points $T(\mathbf{x}_0) = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{p-1}\}$ and is called a *periodic orbit* . The set of periodic points is denoted by

$$P(p) = \{\mathbf{x} \mid \mathbf{f}^{[p]}(\mathbf{x}) = \mathbf{x},\ \mathbf{x} \in \mathcal{M}\}. \tag{2.3}$$

We also consider the subset of $P(p)$ which only consists of points with a least period p :

$$\hat{P}(p) = \{\mathbf{x} \mid \mathbf{x} \in P(p) \text{ and } \forall k < p\ :\ \mathbf{f}^{[k]}(\mathbf{x}) \neq \mathbf{x}\}. \tag{2.4}$$

Obviously, $P(p)$ and $\hat{P}(p)$ are unions of periodic orbits and invariant. Note that for any $p_1 \neq p_2$ the sets $\hat{P}(p_1)$ and $\hat{P}(p_2)$ are disjunct. The set of all points with a period $\leq p$ is denoted by

$$AP(p) = \bigcup_{1 \leq k \leq p} \hat{P}(k). \tag{2.5}$$

In this work we propose methods for the computation of the sets $P(p)$, $\hat{P}(p)$ and $AP(p)$ for a fixed period p.

Mainly for two reasons, the computation of periodic orbits is a nontrivial task which can not be achieved by simple indirect methods, like the application of forward iterations for some initial conditions. Firstly, there might be several coexisting periodic orbits of a size p. For each simulation of an orbit from an initial condition one can find not more than one new periodic orbit. Hence, one

can never be sure if the set of initial points is sufficient to find all coexisting periodic orbits. Secondly, some of the periodic orbits are usually unstable. In more complex systems, we might even find several hundreds or thousands of coexisting unstable periodic orbits. Due to the fact that forward iterations are not suitable to locate unstable invariant sets, unstable periodic orbits can also not be detected by this method.

An approach to overcome the difficulties mentioned above by a direct numerical method was proposed in Nusse and Yorke [NY97]. The focus is put on solving the second problem – the detection of unstable periodic points. In order to do so, the task to find a periodic point of period p is transformed into a root finding problem:

$$\mathbf{f}^{[p]}(\mathbf{x}) - \mathbf{x} = 0. \tag{2.6}$$

Obviously, a real solution (or a root) $\mathbf{x}$ of Eq. 2.6 is also a periodic point. Solving this kind of equation and finding roots by numerical methods is a very well researched field. The standard approach is to use an iteration scheme based on the multidimensional Newton method, see also Chapter 6 and Sec. 7.1 for a more detailed discussion of this topic. However, as is the case for the application of forward iterations, also the application of the Newton method requires an initial value. It depends on this initial value if the Newton method converges to a root and, if so, to which root. In [NY97], a random set of points is proposed to serve as initial values. Consequently, the first problem we mentioned, that one can never know if the initial values are sufficient to find all coexisting periodic orbits is not yet solved. Note that there exist also approaches to find suitable sets of initial values, see Davidchack *et al.* [DLKB01] and references therein, or [BW89, SD97]. However, these techniques can only be applied to some specific scenarios, like for the detection of unstable orbits in chaotic attractors.

It is our intention to propose numerical methods which are capable to detect periodic orbits without the usage of the Newton method. This means that our computation of periodic points does not depend on a set of initial values which must be chosen properly. Furthermore, the Newton method can only be applied if the underlying system $\mathbf{f}$ is differentiable, while our methods only require that $\mathbf{f}$ is continuous.

2.3 The Chain Recurrent Set

The chain recurrent set is of particular interest because it includes all types of return trajectories, like periodic or recurrent orbits. Hence, the computation of the chain recurrent set is a good start for the investigation of a dynamical system.

It provides important global information about a system's structure. Once the chain recurrent set is computed, its components can be considered as subjects of further analysis. Moreover, the chain recurrent set is of importance in context of the Conley index theory, see [Con78, CE71, MM02].

As a first step toward the definition of the chain recurrent set, we introduce the notion of an ε-*orbit*. An ε-orbit can be defined as follows, see also [Ano67].

Definition 2.2. *Fix $\varepsilon > 0$. An infinite sequence $\{\mathbf{x}_k\}$ is called an ε-orbit or a pseudo-orbit of $\mathbf{f}$ if for any $k \in \mathbb{Z}^T$ the distance between the image $\mathbf{f}(\mathbf{x}_k)$ of the point $\mathbf{x}_k$ and the next point $\mathbf{x}_{k+1}$ is less than ε:*

$$\rho(\mathbf{f}(\mathbf{x}_k), \mathbf{x}_{k+1}) < \varepsilon$$

A pseudo-orbit $\{\mathbf{x}_k\}$ is said to be p-periodic if $\mathbf{x}_k = \mathbf{x}_{k+p}$ for each $k \in \mathbb{N}$.

Note that the equality sign is in the above definition admissible. In fact it is important if ε is fixed, and it is not important if ε is arbitrary small, see also [Osi04] and references therein.

A p-periodic ε-orbit (or periodic path) will be denoted by its periodic part $\{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$. The points $\mathbf{x}_k$ are called (p, ε)-periodic. We say that a point is ε-periodic if the period is of no importance.

Note that the concept of pseudo-orbits is especially relevant for numerical computations. Reason for this is that in practice a real orbit is seldom known exactly. What we usually find by numerical methods is nothing more than ε-orbits for sufficiently small positive ε.

Definition 2.3. [Osi04] *A point $\mathbf{x}$ is called chain recurrent if $\mathbf{x}$ is ε-periodic for every positive ε, i.e., there exists a periodic ε-orbit passing through $\mathbf{x}$. The set of chain recurrent points is called a* chain recurrent set *and denoted by Q.*

A chain recurrent set is invariant, closed and contains all types of return trajectories. Among others, these are periodic, quasiperiodic, recurrent, homoclinic and nonwandering trajectories [Osi93]. Although a chain recurrent point is not periodic, it may become periodic under a small C^0-perturbation of $\mathbf{f}$ [Shu87] if the dimension of $\mathcal{M} > 1$.

We consider now the notions of those types of return trajectories which are relevant in the context of this work. Besides periodic trajectories these are mainly quasiperiodic trajectories. We use the following definition.

Definition 2.4. [Has03] *The trajectory* $\mathbf{x}_k = \mathbf{f}^{[k]}(\mathbf{x}_0)$ *is called quasiperiodic if it can be written in the form* $\mathbf{x}_k = \mathbf{G}(k,\ldots,k), k \in \mathbb{Z}^T$ *where the mapping* $(k_1,\ldots,k_m) \mapsto \mathbf{G}(k_1,\ldots,k_m)$ *with* $\mathbf{G} : \mathbb{R}^m \mapsto \mathcal{M}$ *is continuous and periodic with respect to each argument* k_i*.*

By this definition the set of quasiperiodic trajectories comprises the set of periodic trajectories. A quasiperiodic trajectory is periodic if and only if the vector of periods $\mathbf{K} = [K_1,\ldots,K_m]^k$ of the mapping $\mathbf{G}$ is proportional to a vector of integers $\mathbf{n} = \lambda\mathbf{K}$, where the factor λ has to be rational. In case the factor λ is irrational, the quasiperiodic trajectory forms a closed curve [Has03]. For a better distinction, we call those quasiperiodic trajectories with an irrational factor λ truly quasiperiodic trajectories. A truly quasiperiodic trajectory is an m-dimensional manifold if the corresponding map $\mathbf{G}$ requires exactly m arguments. All types of quasiperiodic trajectories are return trajectories and belong to the chain recurrent set.

Besides the localization of truly quasiperiodic trajectories, another main application for the computation of the chain recurrent set is connected with periodic orbits. In the preceding section we introduced $AP(p)$, see Eq. 2.5, as the set of all periodic points with a period $\leq p$. We also mentioned that we developed methods for the computation of these points. By definition, this computation is limited by the fixed least period p. One should consider now that the number p of the least period of periodic points in $\mathcal{M}$ can tend to infinity. In such a scenario, it is impossible to compute the set of all periodic points

$$\bigcup_{p\in\mathbb{N}} AP(p) = AP \tag{2.7}$$

by calculation of a set $AP(p)$. This problem might be solved by computation of the chain recurrent set Q because $AP \subseteq Q$.

A special case to consider occurs if the underlying dynamical system is continuous in time. In such kind of system, a return trajectory is a closed, continuous trajectory, also called a *limit cycle*. The localization of these limit cycles is also an aim of our investigation methods. We mentioned earlier that we do not investigate dynamical systems continuous in time directly, but instead apply a shift operator and transform them into systems discrete in time. In the discretized version of the system, the limit cycles still exist, but they appear as discrete return trajectories. Hereby, it depends on the shift operator by which type of return trajectory the limit cycle of the original system is presented. It might be a fixed point, a periodic orbit, or another type of return trajectory. Due to the fact that all kinds of return trajectories belong to the chain recurrent set, the limit cycles

of the original system can be detected by the computation of the chain recurrent set for the discretized system, no matter which shift operator has been chosen.

Although some parts of the chain recurrent set might be computed by forward iteration, namely those representing stable invariant sets like stable periodic and quasiperiodic trajectories, there is no such simple approach to compute the whole chain recurrent set. To our knowledge, all numerical methods capable of computing an outer covering of the chain recurrent set are, like the symbolic analysis we discuss in this work, considered to be graph-like or set-oriented approaches, see for instance [DJ02] or [Eid95, Mis02]. Reason for this is that there is a natural correspondence – the principle scheme of the chain recurrent set, the ε-trajectory, is similar to the notion of a trajectory in those approaches. Indeed, in symbolic analysis the localization of the chain recurrent set is even considered as a basic computational step which is required for almost all other investigations.

2.4 Attractors, Repellers and Basins

One of the main objectives of dynamical system theory is to describe the final behavior of some evolving states, i.e. the asymptotic behavior of the states as t approaches infinity. The mathematical concept which describes such asymptotic behavior is the attractor. Every trajectory started in a phase space of a dynamical system eventually ends up in an attractor. Hence, the knowledge of a system's attractors is crucial for the understanding of its dynamics. Another important consideration is the question by which attractor a trajectory is attracted or, in other words, what is the basin of attraction of each attractor. If one can locate all attractors of a dynamical system as well as the basins of these attractors, the long term behavior of each initial state could be predicted. This is especially important for the practical implementation of dynamical system theory. If, for example, an engineer knows the attractors and basins of a dynamical system which models a physical process, he can set proper initial conditions to achieve a desired behavior, or he can predict the system's behavior according to the current settings. In a more analytical approach, also repellers are of importance. The repeller is the counterpart of an attractor, and usually its numerical computation is more complicated. However, we present numerical methods to compute all these entities – attractors, their basins and also repellers.

For the investigation of the asymptotic behavior it is convenient to have a general description of all possible asymptotic states of a dynamical system. Such a description is captured in the concept of limit sets, which contain limit points of

orbits. The ω-*limit set* $\omega(B)$ of a set $B \subseteq \mathcal{M}$ is defined as

$$\omega(B) = \bigcap_{k>0} cl\,\mathbf{f}^{[k]}(T^+(B)),$$

where *cl* is the closure. For invertible systems, the α-limit set $\alpha(B)$ is defined analogously as

$$\alpha(B) = \bigcap_{k<0} cl\,\mathbf{f}^{[k]}(T^-(B)).$$

It is easy to see that ω- and α-limit sets are invariant.

Definition 2.5. [BS70, Osi99] *An invariant set* Λ *is an attractor if and only if there exists a fundamental neighborhood* U *of* Λ *so that its* ω*-limit set* $\omega(U) = \Lambda$.

Note hereby that an attractor can be a fixed point, a (quasi-)periodic orbit or some other invariant subset of Q which is chain recurrent. Its structure can be highly nontrivial and fractal. An attractor is called a *strange attractor* if its dynamics are chaotic or, more formally speaking, if it has a non-integer dimension. See also [RT71].

We define the set $W^s(\Lambda)$ as

$$W^s(\Lambda) = \{\mathbf{x} |\ \lim_{n\to\infty} \rho(\mathbf{f}^{[n]}(\mathbf{x}), \Lambda) = 0\}. \tag{2.8}$$

Definition 2.6. [BS70, Osi99] *Let* Λ *be an attractor. The set* $W^s(\Lambda)$ *is called the domain of attraction, or the basin, of* Λ.

Note that the domain of attraction is an invariant set and a neighborhood of Λ [BS70]. Let $\mathbf{x}_0 \in W^s(\Lambda) \setminus \Lambda$ be an initial value in a basin of an attractor Λ, and $T^+(\mathbf{x}_0)$ its trajectory. We denote by $n \in \mathbb{N}$ the position in this trajectory at

$$n = \min(n \mid \forall \mathbf{x}_{k+n} \in T^+(\mathbf{x}_0), k \in \mathbb{N} : \rho(\mathbf{x}_{k+n}, \Lambda) < \varepsilon \text{ for all } \varepsilon > 0).$$

Then, assuming $k \in \mathbb{N}$, this trajectory $T^+(\mathbf{x}_0)$ is considered to be in its transient phase for all $\mathbf{x}_k$ with $k < n$, and in its asymptotic phase for all $\mathbf{x}_{k+n}$. In the context of numerical computations, we say that a forward iteration reaches the asymptotic phase in the k-th step if $\mathbf{x}_k$ is reasonably close to Λ, i.e. $\rho(\mathbf{x}_k, \Lambda) < \varepsilon$. The exact definition of reasonably close depends on the case-specific desired accuracy ε of a computation.

Definition 2.7. [Osi99] *An invariant set* Λ *of* $\mathbf{f}$ *is called a repeller if there exists a neighborhood* U *so that its* α*-limit set* $\alpha(U) = \Lambda$.

We might say that an invariant set Λ is a repeller for $\mathbf{f}$ if Λ is an attractor for $\mathbf{f}^{[-1]}$. Obviously, this definition of a repeller requires that the inverse $\mathbf{f}^{[-1]}$ exists. Hence, it is only valid for invertible dynamical systems, i.e. if $\mathbf{f}$ is a homeomorphism.

Proposition 2.1. [BS70] *An invariant set Λ^* is the dual repeller of Λ if $\Lambda^* = \mathcal{M} \setminus W^s(\Lambda)$.*

The set $\Lambda^* = \mathcal{M} \setminus W^s(\Lambda)$ is a repeller [BS70]. For an invertible system we can say that each trajectory through $W^s(\Lambda) \setminus \Lambda$ begins in the repeller Λ^* and finishes in the attractor Λ. So the pair Λ, Λ^* is called an *attractor-repeller pair*. Our aim is to localize attractors, repellers and their domain of attraction as defined above by numerical methods without any preliminary information about the dynamical system.

Generally, an attractor can be approximated by forward iterations. This can easily be derived from its definition and indeed, the application of forward iterations is still the most straight-forward and well-known method to compute attractors. However, this naive approach is not sufficient for many applications. Reason for this is again the general restriction of this kind of computation – forward iteration can only be applied to a limited set of trajectories, and, due to this limitation, one never knows if all attractors within $\mathcal{M}$ have been detected. Besides that, in the context of filtrations, see Sec. 2.6, we see that there exist invariant sets which can be classified as attractors, though it is only possible to capture them as a whole if the forward iteration was applied for initial values belonging to a very limited range in the phase space. Furthermore, in case an attractor is not a periodic orbit, it is also a difficult, or even impossible, task to decide when a forward iteration reaches the asymptotic phase. And even if the asymptotic phase has been reached, then the forward iterates still cover only parts of the attractor. The same considerations concern also the computation of repellers by backward iteration. Additionally, the computation of backward iterations might be expensive or even impossible if the inverse $\mathbf{f}^{[-1]}$ can not be derived analytically. Note that a different approach for the computation of relative global attractors was presented by Dellnitz and Junge [DH97] which will be discussed in the context of our approaches, see Sec. 5.6.

Similar computational problems as for attractors arise if the basin of attraction is calculated by simulation of orbits. Following this approach, one computes forward iterates whose initial values are spread all over an area $\mathcal{M}$ of investigation. A number n is fixed as the maximal number of iteration steps. In case a forward iteration ends up in (or comes reasonably close to) an attractor Λ after not more than n iterations, the neighborhood of all points belonging to this forward iteration is considered to be part of the basin of Λ. Except that this approach requires

high performance resources, and is therefore usually only applicable for 1- and 2-dimensional systems, two further limitations must be considered. Firstly, it is generally not known if a trajectory already enters the asymptotic phase within n iteration steps. Secondly, even if the trajectory enters the asymptotic phase within those n iteration steps, this might be difficult to find out due to the fact that a test of intersection with the attractor must be performed. In case the attractor has a nontrivial structure and, hence, can only be approximated, this test might be computationally expensive and/or easily fail even if the trajectory is in the asymptotic phase. Different approaches which apply a discretization of phase space were proposed by Hsu [Hsu87] and [NY97]. A more detailed discussion and a comparison with our approach follows in Sec. 5.6.

2.5 Stable and Unstable Manifolds

Stable and unstable manifolds are, like attractors and repellers, fundamental building blocks of dynamical systems. They appear as siblings and are considered to be parts of the "skeleton" of a dynamical system because they indicate the general directions of the system flow. In phase portraits, these manifolds are used as the key trajectories to get a schematic view of the system's dynamics. Besides that, an important property of the stable manifolds is that they indicate the boundaries of basins of attraction. This means that in case the stable manifolds of a system can be computed, the computation of the bounded basins may not be necessary anymore. The advantage becomes clear if we consider that in general the computation of the stable manifolds is more widely applicable and more efficient than the computation of basins. Another important property of stable and unstable manifolds is that their intersection leads to complicated dynamics and chaos, see e.g. [GH83]. Hence, the localization of such an intersection by numerical methods can indicate the occurrence of chaos.

Let $\mathbf{x}_0$ be a fixed point of $\mathbf{f}$. We consider the possibly complex eigenvalues $\lambda_1, \ldots, \lambda_m$ of the Jacobian matrix $D\mathbf{f}(\mathbf{x}_0)$. Each of these eigenvalues lies either inside, on or outside the unit circle $C = \{\lambda \in \mathbb{C} \mid |\lambda| = 1\}$. We say now that $\mathbf{x}_0$ is a *hyperbolic fixed point* if no eigenvalue lies on the unit circle, i.e. $\lambda_n \notin C$ for $n = 1, \ldots, m$. A hyperbolic fixed point is called a *sink* if all eigenvalues are inside the unit circle, a *source* if all eigenvalues are outside, and a *saddle* if there exists both, eigenvalues inside and outside the unit circle.

Let us now assume that $\mathbf{f}$ is a diffeomorphism and has a hyperbolic fixed point $\mathbf{x}_0$ of saddle type. The local stable and unstable manifolds $W^s_{loc}(\mathbf{x}_0)$,$W^u_{loc}(\mathbf{x}_0)$ of $\mathbf{x}_0$

are defined for some neighborhood U of $\mathbf{x}_0$ as

$$\begin{aligned} W^s_{loc}(\mathbf{x}_0) &= \left\{ \mathbf{x} \in U \mid \mathbf{f}^{[k]}(\mathbf{x}) \to \mathbf{x}_0 \text{ as } k \to +\infty, \text{ and } \mathbf{f}^{[k]}(\mathbf{x}) \in U, \forall k \geq 0 \right\}, \\ W^u_{loc}(\mathbf{x}_0) &= \left\{ \mathbf{x} \in U \mid \mathbf{f}^{[k]}(\mathbf{x}) \to \mathbf{x}_0 \text{ as } k \to -\infty, \text{ and } \mathbf{f}^{[k]}(\mathbf{x}) \in U, \forall k \leq 0 \right\}. \end{aligned}$$

The stable manifold theorem [Nit71, Shu87] states that $W^s_{loc}(x_0)$ and $W^u_{loc}(x_0)$ exist on some neighborhood of x_0 tangent to the stable and unstable eigenspaces $E^s(\mathbf{x}_0)$, $E^u(\mathbf{x}_0)$ and that they are of corresponding dimension. Furthermore, $W^s_{loc}(\mathbf{x}_0)$ and $W^u_{loc}(\mathbf{x}_0)$ are as smooth as $\mathbf{f}$.

The global stable and unstable manifolds are defined by taking unions of backward and forward iterates of the local manifolds,

$$W^s(\mathbf{x}_0) = \bigcup_{k \leq 0} \mathbf{f}^{[k]}(W^s_{loc}(\mathbf{x}_0)), \tag{2.9}$$

$$W^u(\mathbf{x}_0) = \bigcup_{k \geq 0} \mathbf{f}^{[k]}(W^u_{loc}(\mathbf{x}_0)). \tag{2.10}$$

We say that $W^s(\mathbf{x}_0)$ and $W^u(\mathbf{x}_0)$ are embedded manifolds in the state space.

These are the classical definitions of the stable and unstable manifolds. Unfortunately, a serious drawback is the underlying assumption that $\mathbf{f}$ is a diffeomorphism. We try now to overcome this restriction and give notions of the stable and unstable manifold according to those in [EKO05] which are valid for a broader class of dynamical systems. See also [MGBC96] for a more detailed discussion of this topic.

Let us consider that the map $\mathbf{f}$ is noninvertible. Obviously, $\mathbf{f}$ is then not a diffeomorphism and multiple inverses might exist. We assume that $\mathbf{f}$ is differentiable in a neighborhood of the fixed hyperbolic point $\mathbf{x}_0$. Let us recall that the global unstable manifold is expressed in terms of the successive union of forward iterates of the local unstable manifold $W^u_{loc}(\mathbf{x}_0)$, see Eq. 2.9. Note that, even if $\mathbf{f}$ is noninvertible, the images of $W^u_{loc}(\mathbf{x}_0)$ will be unique [EKO05]. Hence, $W^u(\mathbf{x}_0)$ is still an embedded manifold in the phase space and we are justified in speaking of an unstable manifold. An alternative definition of the global unstable manifold which avoids the use of the inverse can be given by

$$W^u(\mathbf{x}_0) = \Big\{ \mathbf{x} \in \mathcal{M} \mid \exists \{\mathbf{q}_k\}_{k=0}^{\infty} : \mathbf{q_0} = \mathbf{x} \text{ and } \mathbf{f}(\mathbf{q}_{k+1}) = \mathbf{q}_k, \text{ so that } \lim_{k \to \infty} \rho(\mathbf{q}_k, \mathbf{x}_0) = 0 \Big\}. \tag{2.11}$$

Let us next focus on the global stable manifold. Again, we give an alternative definition of $W^s(\mathbf{x}_0)$. We say that $W^s(\mathbf{x}_0)$ is the set of points that converge to $\mathbf{x}_0$ under forward iteration of $\mathbf{f}$. Hence, $W^s(\mathbf{x}_0)$ can be defined as follows:

$$W^s(\mathbf{x}_0) = \left\{ \mathbf{x} \in \mathcal{M} \mid \lim_{k\to\infty} \rho(\mathbf{f}^{[k]}(\mathbf{x}), \mathbf{x}_0) = 0 \right\}. \tag{2.12}$$

According to this definition, we can still speak of $W^s(\mathbf{x}_0)$ as the stable manifold in case $\mathbf{f}$ is invertible. Recall now that $W^s(\mathbf{x}_0)$ is the union of the successive pre-images of $W^s_{loc}(\mathbf{x}_0)$, see Eq. 2.10. Hence, if $\mathbf{f}$ is noninvertible and multiple inverses exist, then $W^s(\mathbf{x}_0)$ may consist of disjoint pieces [EKO05]. In particular, this set is not an embedded manifold and one also speaks of the *global stable set* instead. Despite that, in the following we will generally refer to $W^s(\mathbf{x}_0)$ as the global stable manifold if the underlying system is a continuous map. Obviously, the definition of the global stable manifold resembles strongly those of the domain of attraction, Eq. 2.8, and indeed, in case $\mathbf{x}_0$ is a sink, we would refer to the global stable manifold as the domain of attraction of $\mathbf{x}_0$. However, the distinction is more than a formal one because, as we have already mentioned, the stable manifold of a saddle has a different meaning regarding the dynamics of a system. Furthermore, the numerical computation generally also requires different approaches.

Note that the concept of stable and unstable manifolds does not only apply to hyperbolic fixed points, but also to hyperbolic periodic orbits. We keep in mind that each point $\mathbf{x}_k$ of a p-periodic orbit is a fixed point for the p-th iterate of $\mathbf{f}$, i.e. $\mathbf{f}^{[p]}$. Thus we have to determine the eigenvalues of the Jacobian

$$D\mathbf{f}^{[p]}(\mathbf{x}_k) = \prod_{n=1}^{p} \mathbf{f}(\mathbf{x}_{p-n+1}).$$

Because cyclic permutations of the matrices within the product do not change the eigenvalues of the matrix, they do not depend on the selection of the periodic point $\mathbf{x}_k$. If a periodic orbit $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ is of saddle type then we say that $W^s(P)$ and $W^u(P)$ are the global stable and unstable manifold. Note that these manifolds are wrapped around each periodic point and, hence, typically consist of disconnected pieces.

In most cases, stable and unstable manifolds can not be acquired analytically, so that it is necessary to apply numerical techniques for their computation. Several approaches already exist to achieve this task, depending on the class of dynamical system to investigate. In case of a dynamical system continuous in time, it is sufficient to provide an algorithm for the computation of the unstable manifold. The stable manifold can then be acquired by following the system's

flow in backward time. Different concepts were developed in order to realize such algorithms. In [GW93] and [KO03] the unstable manifolds are approximated as a sequence of *geodesic circles*. The method of Doedel [DKK91a, DKK91b] computes two-dimensional unstable manifolds by following trajectories B_p as a *boundary value problem*. In a different approach by Henderson [Hen03], the manifolds are constructed by *fat trajectories*, which are trajectories extended by tangent and curvature information at each point. A comprehensive survey about these methods can be found in [KOD+05].

A different scenario occurs if the underlying system is a map. In that case, the above mentioned techniques are in general not applicable because the manifolds do not consist of flows but of sequences of points. Other methods are necessary to compute unstable manifolds. One of them is the approach of Dellnitz *et. al.* [DH97] which approximates an outer covering of the unstable manifold by adaptive subdivision of the state space. As will be discussed later, this method is in many ways similar to the approaches we present in this work. Another technique for the computation of one- and two-dimensional manifolds was introduced by Krauskopf and Osinga [KO98a, KO98b]. Hereby, the computed manifold grows from a local neighborhood of the origin by finding new points on the manifold in a prescribed distance from the last point. Additionally, there exists a popular technique for the computation of one-dimensional manifolds, see [YKY91, HOV95, Sim89, PC89]. In this approach at first the local unstable manifold around its origin is computed, and then a *fundamental domain* is iterated.

All methods mentioned above have in common that they only compute unstable manifolds. In order to get now a stable manifold for a map, these methods must be applied on its inverse. If the inverse exists, but is not available, it might be possible to overcome the problem by numerical approximation using the Newton's method. But in case the inverse does not exist the methods we mentioned so far are insufficient to compute the stable manifolds. In a recent work by England *et. al.* [EKO05] this problem was addressed, and a solution proposed for the computation of one-dimensional stable manifolds. The approach is close to the one in Krauskopf and Osinga [KO98b] for unstable manifolds, and lets the manifold grow by searching for an intersection point of the image of a circle around the last computed point with the part of the manifold that was already computed. Another technique which might be capable to compute one-dimensional stable manifolds for noninvertible maps was proposed in [NY89] but has not been used to compute long pieces of stable manifolds.

In this work we present now methods to compute the global stable and unstable manifolds for continuous mappings in general. Neither the system's inverse nor its

Jacobian is required. We are also not limited to one-dimensional manifolds. Theoretically, the method might be capable to compute manifolds independent of their dimension, though in practice we only applied it to get one- and two-dimensional manifolds. To our knowledge, no other technique is capable to compute higher dimensional stable and unstable manifolds of noninvertible maps.

2.6 Filtrations and Connecting Manifolds

We already mentioned that a dynamical system can have a lot of different attractors and repellers. These components are connected with each other in a complex manner. There might be not only connections between attractor-repeller pairs but also between several layers of attractors and repellers embedded into each other. In order to understand the dynamics of a dynamical system it is important to reveal all of these attractors and their relationships. A concept which is helpful to achieve this task is filtration. It can be seen as a sequence of attractors embedded into each other, and it is our aim to construct such sequences. However, in order to construct attractors it is, besides the calculation of filtrations, also helpful to find out if and how several components of a dynamical system are connected. The links between these components are the so-called connecting manifolds. Their computation is required in order to construct attractors.

We refer here to the concept of filtration as it was proposed in [NS75]. A definition can be given as follows.

Definition 2.8. *A filtration for the homeomorphism* $\mathbf{f}$ *is a finite sequence* $F = \{U_0, U_1, \ldots, U_m\}$ *of open sets so that* $\emptyset = U_0 \subset U_1 \subset \cdots \subset U_m = \mathcal{M}$ *and* $\mathbf{f}(cl\, U_k) \subset U_k$ *for each* $k = 0, 1, \ldots, m$.

Note that a filtration is not unique [NS75]. The next proposition describes the structure of attractors generated by a filtration.

Proposition 2.2. [NS75] *For a given filtration* F *the maximal invariant subset in* U_k, $k = 0, 1, \ldots, m$*:*

$$\Lambda_k = \left\{ \bigcap \mathbf{f}^{[n]}(U_k) \mid n \in \mathbb{Z}^+ \right\},$$

is an attractor for $\mathbf{f}$ *with* $\emptyset = \Lambda_0 \subset \Lambda_1 \subset \cdots \subset \Lambda_m = \mathcal{M}$.

It follows that each U_k is a neighborhood of the attractor Λ_k, and $U_k \subseteq W^s(\Lambda_k)$.

The maximal invariant subset in $U_k \setminus U_{k-1}$ is called a *Morse set* and denoted by

$$K_k(F) = \left\{ \bigcap \mathbf{f}^{[n]}(U_k \setminus U_{k-1}) \mid n \in \mathbb{Z}^T \right\}. \tag{2.13}$$

The set $K_k(F)$ can be considered as the intersection of the attractor Λ_k and the repeller Λ^*_{k-1} dual to Λ_{k-1}. We define $K(F)$ as

$$K(F) = \left\{ \bigcup K_k(F) \,|\, k = 1, \ldots, m \right\}.$$

In [Osi99] it was proved that the chain recurrent set Q lies in $K(F)$. This leads us to the following definition of a *fine filtration*.

Definition 2.9. [NS75] *A filtration is said to be fine if $K(F) = Q$.*

In this work we will show ways how to construct filtrations by numerical methods and identify the attractors Λ_k of the filtration.

In order to achieve this task, it is necessary to consider the components of the chain recurrent set Q and connecting manifolds.

Definition 2.10. [Osi04] *A subset $\Omega \subset Q$ is a component of the chain recurrent set if any two points from Ω can be connected by an ε-orbit for every $\varepsilon > 0$.*

It follows that the chain recurrent set Q can be represented as a union of disjoint closed invariant components:

$$Q = \bigcup_i Q_i.$$

Let $\{Q_i\}$ be the components of Q. We say that there is a connection $Q_i \to Q_j$ between the components Q_i and Q_j if the intersection between $W^u(Q_i)$, Eq. 2.11, and $W^s(Q_j)$, Eq. 2.12, is not empty, $W^u(Q_i) \bigcap W^s(Q_j) \neq \emptyset$. In other words, there exists an orbit which starts in Q_i and ends in Q_j.

Definition 2.11. *Let Q_i and Q_j be two connected components of the chain recurrent set with $Q_i \to Q_j$. We say that*

$$C(Q_i, Q_j) = W^u(Q_i) \bigcap W^s(Q_j)$$

is the connecting manifold of Q_i and Q_j.

To our knowledge, there exists no other adequate method for the numerical computation of filtrations. There are only methods solving parts of the task. In [Mis02, Eid95] a method for the computation of an order relation between chain recurrent components was introduced. Such kind of order relations will also be used by us to build filtrations. Furthermore, in Mischaikow [Mis02] also connecting orbits are studied in the context of the Conley Index Theory, and transition matrices between the chain recurrent components are computed. Several methods exist for the explicit computation of *connecting orbits*, whereby connecting

orbits can be considered as connecting manifolds for dynamical systems continuous in time. One approach for their computation is the solution of a boundary value problem [Bey90, DF89]. To ensure convergence, this requires a good initial guess value for the boundary value solver. Another approach, which is is based on set-oriented investigations and in spirit close to our proposed method, is given by Dellnitz *et. al.* [DJT01]. This method requires the underlying system to be a homeomorphism and also the explicit computation of the inverse. This is not necessary for our method, which can be applied to every continuous mapping.

Chapter 3

The Symbolic Image Graph

We recall that the central concept of symbolic analysis is a directed graph which represents the structure of the state space of the investigated dynamical system. This graph is called the symbolic image and its construction is the basic task for any investigation. Once the system flow has been transformed into such a graph, all further investigation methods can be formulated as graph algorithms. Hence, an efficient implementation of this construction process is crucial for every investigation based on symbolic analysis. In this chapter, we propose algorithms and adequate data structures which are appropriate to achieve this task. We also give a theoretical analysis of the performance. Some of the concepts we mention were already introduced in implementations of related techniques, especially by the GAIO software package [DFJ01], which provides an implementation of several set-oriented methods. We extend these concepts and combine them with new ones in order to give a complete description of all aspects regarding the construction of symbolic image graphs. Additionally, we point out the differences between the two approaches.

We also introduce some investigation methods in order to apply our implementation in practice. Note that we restricted ourselves to basic operations on the symbolic image, namely the localization of chain recurrent sets and of periodic orbits. Reason for this is that the topic of this chapter is not the application of advanced investigation methods, but rather more the provision of a computational framework which allows their easy and efficient integration.

Numerical computations are performed for several dynamical systems in order to verify our theoretical results. We discuss some reasonable parameter settings and the steps to be taken for the acquisition of the data. While doing so, the reader will be introduced to possible fields of application for symbolic analysis.

3.1 Theoretical Background

The symbolic image graph was introduced in Osipenko [Osi83]. We give here a short summary of the theoretical concepts. A comprehensive overview can be found in [Osi04]. Let $\mathbf{f}$ be a continuous mapping on the C^∞-smooth manifold $\mathcal{M}$. Let

$$C = \left\{ M(1), ..., M(n) \,|\, \bigcup_{i=1}^{n} M(i) = M \right\} \tag{3.1}$$

be a finite covering of the area of investigation M by closed sets. We assume that $M = \mathcal{M}$. The sets $M(i) \subset M$ are named boxes of the covering C. For each box $M(i)$ we consider its image $\mathbf{f}(M(i))$ with respect to the flow $\mathbf{f}$ as

$$\mathbf{f}(M(i)) = \{\mathbf{y} \,|\, \mathbf{y} = \mathbf{f}(\mathbf{x}), \mathbf{x} \in M(i)\} \tag{3.2}$$

Then we define for the box $M(i)$ the covering $C(i)$ of its image $\mathbf{f}(M(i))$. This covering consists of boxes $M(j) \in C$, whose intersections with $\mathbf{f}(M(i))$ are not empty :

$$C(i) = \{M(j) \,|\, M(j) \cap \mathbf{f}(M(i)) \neq \emptyset\}. \tag{3.3}$$

Let us construct the directed graph G with n vertices which matches to each box $M(i)$ the vertex c_i. The vertices c_i and c_j are connected by the directed edge $c_i \to c_j$ iff the cell $M(j)$ is an image box of $M(i)$, i.e. $M(j) \in C(i)$. In the following we denote $M(i)$ as a box, and a vertex c_i on G as a cell.

Definition 3.1. *The graph G constructed as described above is called the symbolic image of* $\mathbf{f}$ *with respect to the covering C.*

We can consider the symbolic image as a finite approximation of the flow $\mathbf{f}$, which depends on the covering C. We can change the symbolic image of $\mathbf{f}$ by varying C. If the cells c_i and c_j are connected by the edge $c_i \to c_j$ then there exists a point $\mathbf{x}$ in the box $M(i)$ so that its image $\mathbf{f}(\mathbf{x})$ lies in the box $M(j)$. Denote by $V(G)$ the set of cells on G. The graph G can be considered as a multi-valued mapping $G : V(G) \mapsto V(G)$ between the vertices.

Example 3.1. *The described construction procedure is illustrated by Fig. 3.1. In this example, an area* $M \subset \mathbb{R}^2$ *is covered by boxes* $C = \{M(1), \ldots, M(12)\}$. *Let the image* $\mathbf{f}(M(1))$ *of the box* $M(1)$ *be the area in the center of the picture. Then the covering* $C(1)$ *is given by the set* $\{M(3), M(4), M(6), M(7), M(8), M(10), M(11)\}$. *These boxes are colored gray in Fig. 3.1.(a). So the symbolic image of* $\mathbf{f}$ *with respect to the covering C possesses the edges shown in Fig. 3.1.(b). In order to construct the symbolic image, the covering must be found for all boxes* $M(1)$ $\ldots M(12)$.

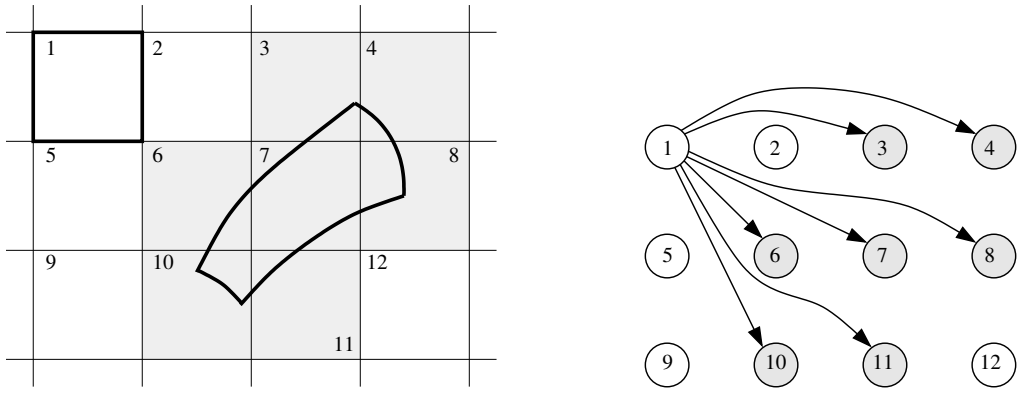


Figure 3.1: *Construction of the symbolic image. (a) The box $M(1)$ is marked with a thick square in the upper left part of the picture. The image $\mathbf{f}(M(1))$ is represented by the area in the center of picture. The covering $C(1)$ is colored in gray. (b) Edges of the symbolic image given by c_1.*

We also introduce some parameters of a symbolic image. Let $\delta(M(i))$ be the diameter of a box belonging to a covering C,

$$\delta(M(i)) = \max(\rho(\mathbf{x},\mathbf{y}) \,|\, \mathbf{x},\mathbf{y} \in M(i)). \tag{3.4}$$

The largest diameter of the cells from C is denoted by

$$\delta(C) = \max(\delta(M(i)) \,|\, M(i) \in C). \tag{3.5}$$

Definition 3.2. *An infinite in both directions (bi-infinite) sequence $\{c_{i_k}\}$ of cells in the graph G is called an admissible path (or simply a path) if for each k the graph G contains the edge $c_{i_k} \to c_{i_{k+1}}$. A path $\{c_{i_k}\}$ is said to be p-periodic if $c_{i_k} = c_{i_{k+p}}$ for each $k \in \mathbb{Z}$.*

A finite path $\omega = \{c_{i_1}, \ldots, c_{i_m}\}$ with the length $|\omega| = m$ is defined on the same way. There is a natural correspondence between the admissible paths on the symbolic image G and the ε-orbits, see Definition 2.2. Roughly speaking, an admissible path represents the trace of an ε-orbit and vice versa.

Definition 3.3. *A cell of a symbolic image is called recurrent if there is a periodic path passing through it. Two recurrent cells c_i and c_j are called equivalent if there is a periodic path containing both, c_i and c_j.*

Denote the subset of recurrent cells in G as $RV(G)$. The set $RV(G)$ decomposes into classes of equivalent recurrent cells

$$H_i = \left\{ c_j \,|\, c_j \in RV(G) \text{ and there is a periodic path containing } c_i \text{ and } c_j \right\}.$$

The sets H_k, each representing a disjunct equivalence class, form together the set

$$\zeta = \{H_1, \ldots, H_n\}\,,\ \forall H_i, H_j \in \zeta, i \neq j:\ H_i \cap H_j = \emptyset,\ RV(G) = \bigcup_{H_k \in \zeta} H_k. \qquad (3.6)$$

In graph theory, a class H_k is called a *strongly connected component* of the graph G. The boxes $M(i)$ belonging to the cells $RV(G)$ are a neighborhood of the chain recurrent set, see Definition 2.3, and the boxes belonging to a set H_k are a neighborhood of a component of the chain recurrent set, see Definition 2.10. As we have already mentioned, the chain recurrent set contains all types of return trajectories. For this reason, the detection of the recurrent cells is the basic task on a symbolic image graph. We will see later that this computation is required as a first step for almost all other investigation methods.

In order to get a better approximation of the vector field $\mathbf{f}$, a multilevel subdivision procedure will be applied as proposed in Osipenko [Osi93]. A similar approach for set-oriented methods exists also by Dellnitz *et al.* [DH96, DJ98]). This procedure can be described as follows. Let C^s be a covering of subdivision level s. Then a subset of boxes belonging to C^s gets selected for the next subdivision. The decision which boxes get selected depends on the kind of investigation which is performed, e.g. if the chain recurrent set should be approximated then all boxes corresponding to recurrent cells of G^s get selected. In general, only the first covering C^0 covers the whole domain M according to Eq. 3.1, and the coverings $C^s, s > 0$ are defined as

$$C^s = \{M(1), \ldots, M(n) \,|\, M(i) \subset M\}\,, \qquad (3.7)$$

and do only cover parts of M. Due to limited memory resources in a practical implementation, it is our intention to cover an area as little as possible by C^s, $s > 0$, and delete all parts of the covering C^{s-1} which are not required for further investigation. Of course, such an approach requires that each C^s is an outer covering of the solution, so that

$$C^0 \supseteq C^1 \supseteq C^2 \supseteq \cdots \supseteq S,$$

where S is the solution.

Definition 3.4. *We denote $M(\infty)$ as the area which is not covered by C^s, i.e.*

$$M(\infty) = \mathcal{M} \setminus \bigcup_{M(i) \in C^s} M(i).$$

In the symbolic image G^s of C^s, $M(\infty)$ is represented by a cell c_∞ with no outgoing edges.

The cell c_∞ can be considered as the target for all edges which point outside of the covering. Note that such a concept is not used in the original works of Osipenko. However, this definition is needed for the practical application of some investigation methods.

Suppose now a new covering C^{s+1} is a subdivision of C^s. The boxes of C^{s+1} are denoted as $m(i,k)$. Each box $M(i)$ which is selected for subdivision will be split up into new boxes $m(i,k), k = 1,2,\ldots$, so that each $m(i,k) \subset M(i)$ and

$$\bigcup_k m(i,k) = M(i). \tag{3.8}$$

Denote by G^{s+1} the symbolic image for the new covering C^{s+1}. In this case the cells of the new symbolic image are designated as $c_{i,k}$ and there is a natural mapping $h : c_{i,k} \to c_i$ from G^{s+1} onto G^s. It holds that

$$h(G^{s+1}(c_{i,k})) \subset G^s(h(c_{i,k})), \tag{3.9}$$

so that every path on G^{s+1} can be transformed on some path on G^s. This means that all paths of G^{s+1} and, respectively, also their corresponding ε-orbits, are included in G^s. However, ε can be fixed to a smaller value for G^{s+1} and so the approximation of the investigated area becomes more precise for each application of the subdivision procedure.

It is obvious that an investigation by symbolic analysis as described above is embedded into the concepts of MPSD. This is due to the fact that, firstly, the coverings C^s, $s \geq 0$, see Eqs. 3.1 and 3.7, are discretizations of the phase space, and, secondly, subdivision is applied on a selected part of the covered phase space after the symbolic image graph has been constructed and investigated.

3.2 Implementation Details

From the viewpoint of an implementation by a computer program, the principle scheme of every investigation according to the above mentioned theoretical concepts can be considered as an iterative process which will be repeated for increasing levels of phase space discretization. At each level the calculation involves three main steps which have to be performed several times:

1. Subdivision of selected parts of the phase space into smaller parts, see Sec. 3.2.1 and Sec. 3.2.3.

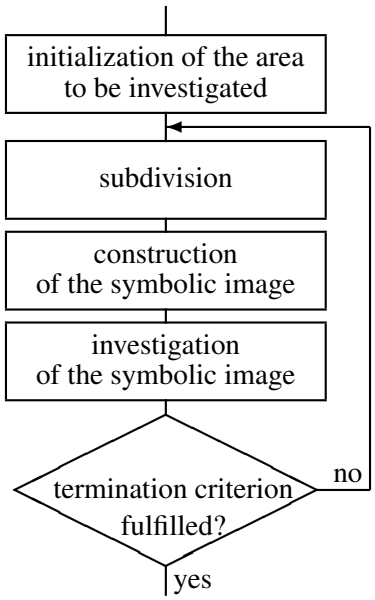


Figure 3.2: Flow chart of the described method by symbolic analysis

2. Construction of the symbolic image for the current discretization of the phase space, see Sec. 3.2.2.
3. Application of an investigation method on the symbolic image graph, see Sec. 3.3. As a result, parts of the phase space get selected for further subdivision and a more precise investigation.

As it is shown in Fig. 3.2, these steps will be repeated until a termination criterion is fulfilled. This condition depends on the desired accuracy as well as on the existing computation power, see Sec. 3.2.3.

In this section we describe the implementation of this basic framework and discuss those aspects which have to be taken into consideration for an efficient implementation. The basic framework as proposed here was implemented and tested within the AnT-project [ant05, ALS+03].

3.2.1 Box and Cell Objects

In order to build a symbolic image for a domain $M \subset \mathbb{R}^d$ of the state space, a finite covering C has to be defined according to Eq. 3.1 or 3.7. In contrast to the theoretical approach, it is usually not possible to cover the complete domain $\mathcal{M}$ of the function $\mathbf{f}$. Instead, in a practical approach we choose an area of investigation $M \subset \mathcal{M}$ in such way that we assume all important dynamics happen inside this

area. Note that M is not necessarily invariant and that usually only those objects can be detected by investigations of symbolic images which are completely covered by M. Generally, there are no restrictions concerning the geometry of M and of the boxes $M(i) \in C$, except that they have to be closed and compact sets. The investigated domain M could be any confined part of the state space and has to be provided by the user. For the simplicity of the implementation we assume here that it is an d-dimensional rectangular region:

$$M = \left[M_1^{\min}, M_1^{\max}\right] \times \cdots \times \left[M_n^{\min}, M_n^{\max}\right]. \tag{3.10}$$

Then we can subdivide the area M into *uniform grid boxes*. In order to do this, the user has to define for each state space coordinate k ($k = 1, \ldots, d$) the numbers $i_k^{\max}$ of subdivisions for the domain M. Then M can be subdivided into

$$m = \prod_{k=1}^{d} i_k^{\max} \tag{3.11}$$

d-dimensional rectangular boxes. The length of the edge k is given for each box by

$$d_k(M(i)) = \frac{1}{i_k^{\max}} \left(M_k^{\max} - M_k^{\min}\right). \tag{3.12}$$

Note that after the application of the subdivision procedure there is no need for the user to explicitly define the domain of the state space which has to be investigated in the $(s+1)$-th step. This domain will be determined automatically, based on the results of the s-th step, see Sec. 3.2.3.

Every box must hold the information about its position in the d-dimensional state space. In context of an efficient implementation, the positions of boxes are represented as d-dimensional *multi-indices* $I \in \mathbb{N}^d$,

$$\begin{aligned} &I = (i_1, \ldots, i_d) \in \mathbb{N}^d \\ &\text{with} \quad i_k \in [1, \ldots, i_k^{\max}], \quad \forall k = 1, \ldots, d, \end{aligned} \tag{3.13}$$

so that for every box $M(I) \in C$ exists a unique multi-index I defining its position in M.

This representation was chosen because it allows a fast and easy mapping from $\mathbf{x} \in M(I)$ to I and vice versa. Furthermore, the mapping range within the domain of usual integer values is much larger if multi-indices are used instead of one-dimensional indices. If N_{int} is defined as the maximal integer value of a

computer system and one would only use one-dimensional indices to identify the boxes $M(i)$ then each $i \in \{1,\ldots,N_{\text{int}}\}$. An area of investigation M could only be subdivided in $m = N_{\text{int}}$ boxes because no more values are available for the description of all possible positions. Note that there must be an index for every position i, no matter if $M(i)$ exists or not. This is a crucial restriction for higher-dimensional systems. If multi-indices are used instead, an d-dimensional domain M can be subdivided in $m = (N_{\text{int}})^d$ boxes which means we have an exponential growth for the number of possible box positions.

Some mapping functions, which will be introduced in the following, require the definition of a *strict weak order* relation $\prec$ for the box indices I of a covering C. It can be given by using the mapping

$$\phi(I) = \sum_{k=1}^{d} \left((i_k - 1) \prod_{l=1}^{k-1} i_l^{\max} \right) \tag{3.14}$$

with $\phi(I) \in \mathbb{N}$ and defining the relation $\prec$ for two indices I and I' by

$$I \prec I' \quad \text{iff} \quad \phi(I) < \phi(I'). \tag{3.15}$$

However, one has to be careful when dealing with the implementation of the relation $\prec$ because for $d \geq 2$ the range of the mapping ϕ defined by Eq. 3.14 can easily overflow the domain of usual integer data types. Therefore, the relation $\prec$ should be implemented without a direct usage of Eq. 3.14. It is better to use an iterative comparison of the components i_k $(k = 1,\ldots,d)$ of an index I, starting with the largest "digit" i_d:

$$\begin{aligned} &I \prec I' \quad \text{iff} \quad \exists k = 1,\ldots,d \\ &\text{with} \quad i_k < i'_k \quad \text{and} \quad i_j = i'_j \quad \forall j > k. \end{aligned} \tag{3.16}$$

After the construction of the covering C is completed, an approximation of the symbolic image based on C can be constructed. It represents a graph G, whereby the cells c_I of G correspond to the boxes $M(I)$ of C. Each of the cells has an *adjacency list* with its target cells. Note that we do not use an *adjacency matrix* to define the edges of the graph G. Reason for this is that the symbolic image graph is considered to be huge but sparse. Hence, an adjacency matrix would require by far more memory resources than a list. A distinctive feature of the cells is, that each of them is uniquely connected with a box $M(I)$. This correspondence represents the link between the domain M and the symbolic image G.

Additionally, we add the cell c_∞ to the graph G which corresponds to the area $M(\infty)$, see Def. 3.4. Hereby, $M(\infty)$ covers the area $\mathcal{M} \setminus M$ as well as the area $M \setminus \bigcup_{M(i)\in C} M(i)$ which is not further investigated. In case $\mathbf{f}(M(i)) \cap M(\infty) \neq \emptyset$, we assume that $M(\infty) \in C(i)$.

3.2.2 Construction of the Symbolic Image

The construction of a symbolic image based on numerical calculations is always only an estimation of the "real" symbolic image G. Besides the usual numerical errors which occur by the computation of a mapping $\mathbf{f}(\mathbf{x})$ for $\mathbf{x} \in M$, another reason for this is the fact that the construction of the image

$$\begin{aligned} T(I) &= \mathbf{f}(M(I)) \\ &= \{\mathbf{y} \mid \mathbf{y} = \mathbf{f}(\mathbf{x}), \mathbf{x} \in M(I)\} \subset \mathbb{R}^d \end{aligned} \tag{3.17}$$

for a box $M(I)$ would involve the calculation of $\mathbf{f}(\mathbf{x})$ for every $\mathbf{x} \in M(I)$. This is, of course, beyond the limits of every finite numerical computation.

One approach for the approximation of $T(I)$ is given by the usage of *interval arithmetic*, see for instance [AH83]. This technique was used by Hruska [Hru02] in the context of box chain construction. The basic idea is to perform numerical computations on intervals instead of numbers. The results of such computations are then again intervals which include all solutions. In higher dimensions, the operations can be carried out component-wise on *interval vectors*. A box $M(I)$ can then be seen as a higher-dimensional interval, and $\mathbf{f}$ can be computed using interval arithmetics. As a result, we get an outer covering of $T(I)$.

Although this is an interesting approach, we did not consider it to be appropriate for our implementation. Reason for this is, that the error bounds easily tend to increase largely, as was also reported in [Hru02]. This would lead to an increase of edges in G which is not desirable. Concerning this topic, see also the discussion about the method's performance and its tuning in Secs. 3.4 and 4.2. Additionally, the system function $\mathbf{f}$ is limited to those operations which are defined for interval arithmetic.

So, for our implementation we use a different method. The image $T(I)$ will be approximated by a finite set of points. This technique was also used by [Hsu87, DH97], and has proved to be a good approach in practice. From each box $M(I)$ a representative set of k points is selected,

$$S(I) = \left\{\mathbf{x}_j \mid \mathbf{x}_j \in M(I), \quad j = 1 \ldots k\right\} \tag{3.18}$$

the so-called *scan points* of the box $M(I)$. Then the approximation $\tilde{T}(I)$ of the region $T(I)$ in the state space is calculated by

$$\tilde{T}(I) = \mathbf{f}(S(I)) = \left\{\mathbf{y}_j \mid \mathbf{y}_j = \mathbf{f}(\mathbf{x}_j), \mathbf{x}_j \in S(I)\right\} \tag{3.19}$$

As one can see, the continuous region $T(I)$ will be approximated by the discrete set $\tilde{T}(I) \subset T(I)$ consisting of k points. The number k of scan points for the boxes

as well as the positions of these points within the boxes are parameters of the described method which must be set by the user.

There is no general strategy how the scan points should be placed within the box $M(I)$. In Dellnitz *et. al* [DH97] it was proposed that the points should lie on the boundary or on the edges of the box. Additionally, there should be one point in the center of the box. This strategy should not be used for symbolic images. One has to consider that the boundaries of the boxes overlap. Hence, if the scan points of the boxes lie on the boundary, they also overlap. This leads to the occurrence of clusters of boxes during subdivision. A better strategy is that the scan points are either uniformly distributed within $M(I)$ or that they are put into the neighborhood of the boundaries.

Besides the calculation of scan points, a mapping

$$p : M \mapsto \mathbb{N}^d, \quad \forall \mathbf{x} \in M(I) \;\Rightarrow\; p(\mathbf{x}) = I \tag{3.20}$$

of a point $\mathbf{x} \in M(I)$ onto a box index I is required for further computations. Additionally, we need its inverse mapping

$$p^{-1} : \mathbb{N}^d \mapsto M, \quad \forall I : \quad p^{-1}(I) = \mathbf{x} \;\Rightarrow\; \mathbf{x} \in M(I) \tag{3.21}$$

which defines for every $M(I)$ the spatial coordinates of a point within this box. Note that $p^{-1}(I)$ is only defined if $M(I)$ exists for I.

Due to the fact that all uniform grid boxes have the same size, the mapping $p : M \mapsto \mathbb{N}^d$ can be simply defined as

$$\begin{aligned} &p(\mathbf{x}) = I = (i_1, \ldots, i_d) \\ &\text{with } i_k = \left\lfloor \frac{\mathbf{x}_k - M_k^{\min}}{i_k^{\max}} \right\rfloor + 1, \quad k = 1, \ldots, d \end{aligned} \tag{3.22}$$

The inverse mapping can be described in a similar way. Note, however, that the inverse mapping is not unique and, in practice, requires the definition of an arbitrary reference point within the box $M(I)$. If using, for instance, the minimal point of each box, one can get

$$\begin{aligned} &p^{-1}(I) = \mathbf{x} = (x_1, \ldots, x_d)^T \\ &\text{with } x_k = M_k^{\min} + (i_k - 1) \cdot d_k(M(I)), \\ &k = 1, \ldots, d \end{aligned} \tag{3.23}$$

After the functions p and p^{-1} have been defined, the approximation $\tilde{C}(I)$,

$$\tilde{C}(I) \;=\; \left\{ M(I') \mid M(I') \cap \tilde{T}(I) \neq \emptyset \right\}, \tag{3.24}$$

of the covering $C(I)$,

$$C(I) \quad = \quad \left\{ M(I') \mid M(I') \cap T(I) \neq \emptyset \right\}, \tag{3.25}$$

can be computed. Obviously, we have the relation $\tilde{C}(I) \subseteq C(I)$.

Proceeding this task, the following steps have to be performed for each box $M(I)$ in the covering C :

1. The set of scan points $S(I)$ is calculated using a set of k globally defined relative coordinates

$$S = \Big\{ \xi_j \mid \quad \xi_j = (\xi_{j,1}, \ldots, \xi_{j,d})^T, \quad \xi_{j,i} \in [0,1], i = 1..d, j = 1..k \Big\} \tag{3.26}$$

with respect to the reference point $\mathbf{x}_0 = p^{-1}(I)$:

$$S(I) = \{ \mathbf{x}_j \mid \quad x_{j,i} = x_{0,i} + \xi_{j,i} \cdot d_i(M(I)), i = 1..d, \quad \xi_j \in S, j = 1..k \} \tag{3.27}$$

This is necessary for scaling the relative coordinates ξ_j, which are defined within the hypercube $[0,1]^d$, onto the area $M(I)$.

2. For every point $\mathbf{x}_j \in S(I)$ the target point $\mathbf{y}_j \in \tilde{T}(I)$ will be calculated.

Dealing with dynamical systems discrete in time, i.e. $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k)$, the target point $\mathbf{y}_j$ can be found by a simple one step iteration of the point $\mathbf{x}_j$:

$$\mathbf{y}_j = \mathbf{f}(\mathbf{x}_j) \tag{3.28}$$

Note that also the n-th iterated function $\mathbf{f}^{[n]}$ can be used in Eq. 3.28 instead of $\mathbf{f}$. This is necessary if we want to apply symbolic images to dynamical systems continuous in time, see Sec. 4.1.1, or if we want to work with higher iterated functions, see Sec. 4.2 for a more detailed discussion.

3. For each target point $\mathbf{y}_j \in \tilde{T}(I)$ the corresponding box object $M(I')$ with $\mathbf{y}_j \in M(I')$ must be found.

The index I' of this box is given by

$$I' = p(\mathbf{y}_j) \tag{3.29}$$

It is important to check, whether the conditions

$$1 \leq i'_k \leq i_k^{\max} \quad \forall k = 1, \ldots, d \tag{3.30}$$

hold for the components of the multi-index I'. If not, the index I' exceeds the dimension range and there is no box defined for this index. In such a case, $\mathbf{y}_j \in \mathcal{M} \setminus M$. Hence, $M(\infty)$ is added to the list $\tilde{C}(I)$.

4. Within the implementation context, a memory access function

$$\begin{aligned} &\mathbf{g} : \mathbb{N}^d \mapsto M, \\ &\exists M(I') \in C \;\Rightarrow\; \mathbf{g}(I') = M(I') \end{aligned} \tag{3.31}$$

is required in order to access a box object $M(I')$ for an index I'.

One can consider at least two approaches to define $\mathbf{g}$:

(a) There is an array $A = [1, i_1^{\max}] \times \cdots \times [1, i_d^{\max}]$ covering all possible index positions I in M and containing pointers to every box $M(I)$ if such an $M(I)$ exists.

(b) There is a *hash map* H which contains a key I iff there exists a corresponding $M(I)$.

The first approach, the use of an array, would be very fast and allows the detection of $M(I)$ in constant time. Unfortunately, it also requires a large amount of memory space. For every possible index position I in M memory is needed, even though there might be no box objects $M(I)$ at many index positions. This aspect becomes more crucial for every subdivision step and would not allow a precise calculation of non-trivial symbolic images. Therefore, the second approach should be preferred.

A hash map H is used to map I onto $M(I)$ if such a box object exists. No memory space is wasted for indices without a corresponding box object. It should be mentioned that a fast and proper implementation of H requires the strict weak ordering $\prec$ of I mentioned before, see Eq. 3.16.

5. If $M(I')$ has been located, a reference to it will be added to the list of $\tilde{C}(I)$. If no $M(I')$ exists at the position I', then $M(\infty)$ is added to $\tilde{C}(I)$. In such a case, $\mathbf{y}_j \in M \setminus C$. When the location of the target boxes $M(I')$ for all scan points $\mathbf{x}_j \in S(I)$ is completed, the list of $\tilde{C}(I)$ is an estimation of the covering $C(I)$ for the box object $M(I)$.

Example 3.2. *Fig. 3.3 illustrates the construction of a covering* $\tilde{C}(1)$ *for a box* $M(1)$. *As one can see, the scan points* $\mathbf{x}_j$*, marked with* $\circ$*, are mapped onto the*

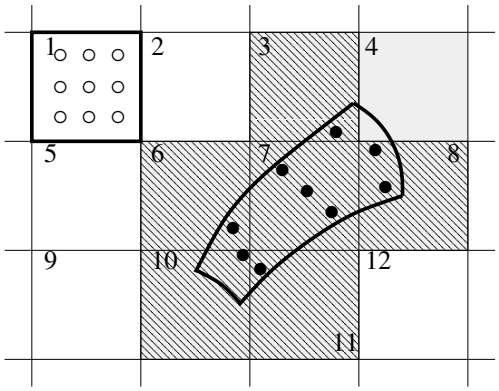


Figure 3.3: *Implementation of the symbolic image construction. The scan points* $\mathbf{x}_j \in S(1) \subset M(1)$ *are marked with* $\circ$*. The images of these points* $\mathbf{y}_j \in \tilde{T}(1) \subset C(1)$ *are marked with* $\bullet$*.*

points $\mathbf{y}_j$*, marked with* $\bullet$ *(*$j = 1..9$*). Based on these points, the following approximation* $\tilde{C}(1)$ *of the covering* $C(1)$ *was calculated:*

$$\tilde{C}(1) = \{M(3), M(6), M(7), M(8), M(10), M(11)\}. \tag{3.32}$$

In Fig. 3.3 the covering $C(1)$ *is colored gray and the approximation* $\tilde{C}(1)$ *is shown as a hatched area. As one can see, the box* $M(4)$*, which belongs to* $C(1)$*, gets lost in the approximation* $\tilde{C}(1)$*, so that* $\tilde{C}(1) \neq C(1)$*. This is is due to the insufficient number of scan points, a typical problem which might occur in numerical simulations with a discretization of the scan points S.*

Note that there exists a method [Jun00] for a *rigorous* computation of the sets $C(I)$ by scan points. This method was introduced for the application on set-oriented methods but could be used in our context as well. If the Lipschitz constants can be estimated for $\mathbf{f}$ on M then this method is able to compute a set $\tilde{C}(I)$, so that $C(I) \subset \tilde{C}(I)$. Though this technique has not yet been implemented by us, we introduce in Sec. 4.1.2 an error tolerance which can be seen as a basic framework for the integration of this technique. However, one should consider that the application of this method, like the interval arithmetic mentioned earlier, provides an outer covering of $C(I)$. Heuristic experience has shown that such an approach for the approximation of $C(I)$ is often too pessimistic, and in many cases not desirable for practical calculations. See the results of Secs. 3.4 and 4.2 for a more detailed discussion.

After the described steps were performed for all boxes of the state space discretization, an approximation G of the symbolic image has been constructed. The

vertices of the graph are the cells c_I corresponding to the boxes $M(I)$. For each c_I the adjacency list of target cells is given by the cells corresponding to the boxes of the covering $\tilde{C}(I)$.

3.2.3 Subdivision Process

In the previous section, the construction of a symbolic image G was described. It was already mentioned that the precision of the state space discretization for such a symbolic image must be increased by an iterative process. To describe this process, we introduce an index s which indicates the level of state space discretization, $s = 0, 1, \ldots$, or, in other words, the subdivision depth of a symbolic image. In the following, the notation for the symbolic image G is extended to G^s. The notation C^s, I^s and so on is introduced in the same way. After G^s is constructed, it is necessary to decide if the process has to be continued in order to construct the next image G^{s+1}, or if the construction process should be stopped, see Fig. 3.2. There are two typical reasons to terminate the process:

- The user-defined maximal number of subdivision steps has been reached. This should be the normal case and means that the symbolic image G^s was calculated with the desired accuracy.
- Although the maximal number of subdivision steps has not yet been reached, the number of cells in the symbolic image G^s is so huge that the next subdivision of cells would overflow the memory space of the used computer. This can be interpreted as some kind of failure. An appropriate output will be produced by the software and the next subdivision will not be performed.

If the subdivision process will be continued in order to get the graph G^{s+1} then a new covering C^{s+1} has to be calculated. This covering depends on a selection of cells $SV(G^s) \subseteq V(G^s)$ chosen by the application of an investigation method to the graph G^s. The covering C^{s+1} usually covers only parts of the area covered by C^s. Let c_{I^s} be the cell in G^s which matches to the box $M(I^s)$. Then the new area of investigation is given by the joint of all boxes which belong to a selection of cells $SV(G^s)$:

$$M^{s+1} = \bigcup_{c_{I^s} \in SV(G^s)} M(I^s). \tag{3.33}$$

Each of these boxes will be divided into m sub-boxes $M(I^{s+1})$, see Eq. 3.11, which build together the new covering C^{s+1}. For every $I^s = (i_1, \ldots, i_d)$ the m new indices

I^{s+1} are defined as follows:

$$\begin{aligned} I^{s+1} &= \Big(j_1 + (i_1 - 1) \cdot i_1^{\max}, \ \dots \ , \\ &\qquad j_d + (i_d - 1) \cdot i_d^{\max}\Big) \\ \text{with} &\qquad j_k = 1, \dots, i_k^{\max} \quad \forall k = 1, \dots, d \end{aligned} \tag{3.34}$$

After the subdivision, the graph G^{s+1} is constructed for the covering C^{s+1} as described in Sec. 3.2.2, and the whole calculation process is repeated.

3.2.4 Comparison with a Similar Implementation

In order to summarize the description of the implementation details, we compare our approach with those of others. To our knowledge, there exists only one implementation which is comparable with symbolic image construction, and for which details were published. Namely, this is the implementation by Dellnitz *et. al.* [DH97, DJ02] for set-oriented methods. Note that the concepts of the set-oriented methods differ from those of symbolic analysis. Roughly speaking, set-oriented methods apply investigations on sets and not on graphs. However, both concepts require the discretization of the phase space, a multilevel subdivision scheme and the computation of the image of a set, i.e. $\mathbf{f}(M(i))$. Other authors, like Mischaikow [Mis02], also refer to this implementation as the basic framework for the application of their methods. Another implementation was introduced by Hruska [Hru02, Hru05] but only a few details are given about those aspects of the implementation which we described here.

For the comparison of our implementation with the one of Dellnitz *et. al.*, we focus on aspects of storage, subdivision and mapping. As mentioned before, the selection of scan points is similar in both implementations. The main difference is that the approach of Dellnitz *et. al.* uses a *binary search tree* for the storage of the collection of sets (or boxes in our terminology). This tree spans over the elements of all subdivision steps. An indexation of the sets is not explicitly given. Instead, it is implicitly determined by a set's position in the binary tree. In order to reach a set belonging to a covering C^k, one has to start at the initial covering C^0 and then traverse the binary tree downwards through the coverings $C^1, \dots, C^k$. As in our approach, the coordinates of a box will not be stored but calculated if needed.

In our opinion, this approach has several disadvantages in comparison with an explicit indexation and the storage in a hash map as it is proposed by us:

Flexibility A binary search tree dictates a special geometric structure for the boxes. The basic covering is a rectangle, and in every subdivision this rectangle can only be subdivided into two pieces. In our approach, one can use uniform grid boxes at every stage of subdivision and divide them into as many parts as desired. So it is possible to integrate several subdivision steps on the binary search tree into one. Furthermore, the geometry of a box is not bounded to its storage scheme. This makes it possible to easily extend the implementation on other geometric forms, or even "intelligent" boxes which decide their geometry themselves. The explicit indexation allows in general a larger degree of freedom.

Memory requirements The binary tree needs to store boxes of all subdivision steps, while in our approach, all boxes except those of the current subdivision step are deleted.

Performance In order to find the corresponding box object to a value $\mathbf{f}(\mathbf{x})$, one must traverse the binary tree. During this traversal, the coordinates of every visited box must be calculated. This operation gets more expensive the more subdivision steps must be traversed. In our approach, the coordinates of a box are only calculated once – to compute the index of a box belonging to a value $\mathbf{f}(\mathbf{x})$, see Eq. 3.20. The search on the hash map in order to find the box for this index does not involve the computation of further coordinates, see Eq. 3.31. Note hereby that the hash map might also be represented by a binary tree. However, this tree is only spanned over the elements of one subdivision step in contrast to the binary tree of Dellnitz *et. al.* which is spanned over the elements of all subdivision steps.

3.3 Basic Investigations

In the preceding section we described the construction of the symbolic image graph. In this section, we discuss basic investigation techniques that can be applied to the symbolic image graph in order to analyze properties of the underlying dynamical system. Note that we use variations of some standard graph algorithms. A discussion of the original algorithms is out of scope of this work, therefore we refer to [AHU87, Tar91, Sed93, CLR00].

3.3.1 Localization of the Chain Recurrent Set

The most important kind of investigation technique on the graph is the localization of the recurrent cells, see Def. 3.3. Applying this technique, we can determine a

neighborhood of the chain recurrent set Q, see Definition 2.3. Let us denote such a neighborhood as

$$QS^s = \{\cup M(i) \,|\, c_i \in RV(G^s)\},$$

where s is the subdivision depth. Recall that a $M(i) \subset M$ is the box corresponding to the cell c_i of the symbolic image G^s, and $RV(G^s)$ is the subset of the recurrent cells in G^s. Obviously, this neighborhood is an outer covering of the chain recurrent set. Furthermore, recall that $\delta(C^s)$ is the largest diameter of the covering C^s, see Eq. 3.5. In our implementation, $\delta(C^s)$ depends on the length of the box edges, see Eq. 3.12.

Theorem 3.1. [Osi94] *The sequence of sets $QS^0, QS^1, QS^2, \ldots$ offers the following properties:*

1. *the neighborhoods QS^s are embedded into each other, i.e.,*

$$QS^0 \supseteq QS^1 \supseteq QS^2 \supseteq \cdots \supseteq Q$$

2. *if the largest diameter $\delta(C^s) \to 0$ as s tends to infinity then*

$$\lim_{s\to\infty} QS^s = \bigcap_s QS^s = Q.$$

This theorem states that the chain recurrent set can be approximated as precisely as one likes by the methods of symbolic analysis. The symbolic image must be constructed and subdivided for several times. The cells which have to be selected for subdivision are those which are recurrent. Hence, we propose now an algorithm for the localization of the recurrent cells in a symbolic image. Note that almost all other investigation methods of symbolic analysis also require the detection of the recurrent cells. Therefore, this technique is considered as a general first computation step of all investigations on a symbolic image graph.

Remark 3.1. *In the original work [Osi94] a homeomorphism is assumed as the underlying dynamical system for Theorem 3.1. However, the theorem is also valid for noninvertible mappings. Reason for this is that the inverse of the system function is not required for its proof. Hence, we assume that the proposed method can be applied on all dynamical systems generated by continuous mappings, no matter if an inverse exists or not.*

An efficient approach to detect the recurrent cells is the variation of Tarjan's algorithm for the calculation of strongly connected components in directed graphs [Tar72]. This algorithm locates the strongly connected components of a

directed graph G by a *depth-first search*. Two vertices a and b of G are said to be strongly connected ($a \sim b$) if there exists a path from a to b and from b to a. Furthermore, the relation $a \sim a$ (reflexivity) always holds by definition. It can easily be proved that $\sim$ is an equivalence relation and that therefore G will be partitioned by the relation $\sim$ into equivalence classes, the strongly connected components. Although recurrent cells of G and strongly connected components are not the same, they are closely related to each other. If

$$\gamma_a = \{b \mid a \sim b\} \tag{3.35}$$

is a strongly connected component for which there is a path from a to each b and vice versa with $a \neq b$ then, for a as well as b, exists a periodic path, see Def. 3.2. It follows that, if $|\gamma_a| > 1$, then for all cells $c \in \gamma_a$ exists a periodic path and therefore all these cells are recurrent. The special case to look at is $|\gamma_a| = 1$. Due to reflexivity, if there is only one component in the set it could mean that this cell is either non-recurrent or, if there is an edge $a \rightarrow a$, its least period size is 1.

So, for the localization of recurrent cells, Tarjan's algorithm needs a minor extension. What has to be done in addition is to perform a test for each set γ_a if $|\gamma_a| = 1$ holds. In this case, it has to be checked for the single cell of this set whether it is one-periodic (or recurrent), which means one of its target cells is itself, or not. If the cell is not one-periodic, it is non-periodic (or non-recurrent). All cells belonging to a set γ_a with $|\gamma_a| > 1$ are periodic, i.e. recurrent. All recurrent cells that belong to the same set γ_a can be considered as one of the equivalence classes $H_k \in \zeta$, i.e. as a set of equivalent recurrent cells, see Def. 3.3 and Eq. 3.6.

We like to mention here that the approach to localize the chain recurrent set by computation of strongly connected components is standard, and was also used by other authors. In [Eid95, Mis02] a slightly different relation between strongly connected components and chain recurrent sets was drawn. However, the resulting algorithm, like Tarjan's, is also based on a depth-first search and in $O(n)$. In [DJ02] the exact algorithm was not outlined but has also the same performance properties.

3.3.2 Localization of Periodic Points

A related investigation is the localization of p-periodic points $P(p)$ for a given value p, see Eq. 2.3. Let us denote the neighborhood of $P(p)$ as

$$SP^s = \{\cup M(i) \,|\, c_i \text{ is } p\text{-periodic and } \in V(G^s)\}\,,$$

where s is the subdivision depth. Obviously, this neighborhood is an outer covering of $P(p)$.

Theorem 3.2. [Osi93] *The sequence of sets $SP^0, SP^1, SP^2, \ldots$ offers the following properties:*

1. *the neighborhoods SP^k are embedded into each other, i.e.,*

$$SP^0 \supseteq SP^1 \supseteq SP^2 \supseteq \cdots \supseteq P(p)$$

2. *if the largest diameter $\delta(C^s) \to 0$ as s tends to infinity then*

$$\lim_{s\to\infty} SP^s = \bigcap_s SP^s = P(p).$$

This theorem states that the set of p-periodic points can be approximated as precisely as one likes by the methods of symbolic analysis. The symbolic image must be constructed and subdivided for several times. The cells which have to be selected for subdivision are those which are p-periodic. Hence, we propose now an algorithm for the localization of the periodic cells in a symbolic image.

Remark 3.2. *In the original work [Osi93] a homeomorphism is assumed as the underlying dynamical system for Theorem 3.2. However, the theorem is also valid for noninvertible mappings. Reason for this is that the inverse of the system function is not required for its proof. Hence, we assume that the proposed method can be applied on all dynamical systems generated by continuous mappings, no matter if an inverse exists or not.*

For a practical application of Theorem 3.2, we have to consider that there might be more than one admissible path to which a cell c_i belongs to, especially in case of a coarse phase space discretization. Indeed, the number of admissible paths to which a cell belongs can even be infinite. In that case, it is impossible to explicitly compute each periodic path to which the cell belongs. On the other hand, if each cell of a symbolic image represents exactly one periodic point in the state space then this cell c_p belongs only to one p-periodic path $\{\ldots, c_{i_0}, \ldots\}$ with $c_{i_0} = c_p$. Although such a precise covering can not be achieved by numerical computation, a reasonably fine phase space discretization is usually sufficient to get a unique path for a cell which belongs to a covering of a periodic point. Considering these facts, the p-periodic points can be located by selecting all cells for subdivision which have a *shortest periodic path* for a period $p' \leq p$. Obviously, such a selection contains all cells which belong to a p-periodic path. After several subdivisions, we check that there exists a unique periodic path for each cell of the symbolic image. In case this can not be achieved, a higher precision of the

symbolic image is required, i.e. more subdivisions must be applied.

Consequently, an algorithm is needed which is able to find the shortest periodic path $c_i \rightarrow \cdots \rightarrow c_i$ for every recurrent cell c_i on the symbolic image graph. Furthermore, the length of such a path must be detected. Note that Tarjan's algorithm is not capable to solve this task. Instead, we introduce a different algorithm which is based on the idea of Dijkstra's algorithm for calculation of shortest paths in directed graphs [Dij59]. It belongs to the class of so-called *greedy algorithms* and performs a *breadth-first search*.

The Dijkstra algorithm does not only find the shortest paths from each cell c_i to all other cells of the graph but also locates at the k-th step first the path $c_i \rightarrow \cdots \rightarrow c_u$ so that the following equation is fulfilled:

$$d(c_i, c_u) = \min\{d(c_i, c_v) \quad | \quad c_v \in V(G) \wedge (c_i \rightarrow \cdots \rightarrow c_v) \notin D_{k-1}\}, \tag{3.36}$$

where $d(c_i, c_u)$ is defined as the length n of the shortest path between c_i and c_u, and D_{k-1} is the set of all shortest paths which have already been detected in the previous steps. Then the shortest periodic path of a cell c_i can be found by checking for the first detected shortest path $c_i \rightarrow \cdots \rightarrow c_u$ whether the edge $c_u \rightarrow c_i$ exists. If so, the algorithm can be stopped because the path $c_i \rightarrow \cdots \rightarrow c_u \rightarrow c_i$ is the shortest periodic path for the cell c_i and the length of this path is the period of c_i. If not, then the next shortest path has to be detected and checked for the same condition until a periodic path has been found or until all shortest paths have been visited.

There are several improvements to speed up Dijkstra's algorithm within our context. First of all, the original Dijkstra algorithm is developed for weighted graphs while the edges of G are unweighted. This means that the edge weight $\gamma(c_i \rightarrow c_j)$ is 1 for all edges of G. Therefore, the outer edge of visited but not yet examined cells can be implemented as a queue. Every cell which is visited first time and becomes a part of the outer edge will be pushed into the queue, while the next cell which will be examined can be popped out of the queue. This works fine because our edges are unweighted and so the distance between c_1 and the first element in the queue is always the minimum distance between c_i and every other element in the queue.

Next it should be considered that all periodic cells of G have to be inspected. So in worst case, the modified Dijkstra algorithm must be started once for each cell $c_i \in V(G)$. In order to spare out some of the cells, we can first run the

Tarjan algorithm to detect the recurrent cells and the sets ζ of equivalent recurrent cells. The Dijsktra algorithm must then only be started for the recurrent cells $c_i \in RV(G)$. Furthermore, it is sufficient to check for each of these cells only the paths to equivalent recurrent cells, i.e. those cells which belong to the same set $H_k \in \zeta$. Cells which do not belong to the same set can not belong to the same shortest periodic path.

Despite all improvements, the modified Dijkstra's algorithm can not compete with the performance of the aforementioned Tarjan's algorithm. So it should only be chosen by the user if the additional information about the periodic paths and/or the least period sizes are really required for the calculation. Note that we will also propose an improved approach for the localization of periodic points in Section 7.1.

The idea of using a breadth-first search algorithm like Dijkstra's for investigations on a symbolic image-like graph is not new, see for instance [Hru02, Mis02]. However, to our knowledge it was not yet modified and improved in order to apply it for the detection of shortest periodic paths.

Another option to compute shortest periodic paths on a graph is given by the Floyd-Warshall algorithm [Flo62, CLR00]. This algorithm uses the principles of *dynamic programming* and solves the all pairs shortest path problem. As with Dijkstra's, this algorithm could easily be modified to compute the shortest periodic paths, see also the method proposed in [Jun03]. For several reasons, we consider this approach not to be advantageous. First of all, this algorithm works on the adjacency matrix of the graph, while we use an adjacency list for the storage of edges. As we have already mentioned in Sec. 3.2.1, the storage of edges in an adjacency matrix would immensely increase the memory requirements. Furthermore, the time complexity of the Floyd-Warshall algorithm is $\Theta(n^3)$. Even if modified for our problem, this complexity will not change. On the other hand, we show in the next section that our approach has a significantly better time complexity of $O(n^2)$ if the number of edges in the graph is fixed, i.e. the construction depends on a fixed number of scan points per box.

3.4 Performance Analysis

The performance of symbolic image construction as described above will be analysed by studying the *worst-case scenario*. In this section we show that the construction of a symbolic image, as well as the basic operations on it, can be done in the time $O(n_s \log(n_s))$, see Proposition 3.2. In the following, n_s denotes

the number of cells in the symbolic image G^s.

In order to construct G^s, we consider a function *getBoxMapping*($M(I)$) which is called for every box $M(I) \in C^s$. This function calculates an estimation $\tilde{C}(I)$ for $C(I)$ by first locating all indices I' with $M(I') \in \tilde{C}(I)$ and afterwards accessing these boxes by calling the function $\mathbf{g}(I')$, see Eq. 3.31. We discuss the performance of these steps in the following.

Remark 3.3. *The localization of the box index I' for the box $M(I')$ with $M(I') \in \tilde{C}(I)$ is in $O(1)$.*

In order to find an index I'_i with $i = 1..k$ and k is the number of scan points, first for each point $\mathbf{x}_i \in S(I)$ the value $\mathbf{y}_i \in \tilde{T}(I)$ has to be processed. The calculation of $\mathbf{y}_i$ requires only the calculation of the system function $\mathbf{f}$ for m_f times and is therefore $\in O(m_f \cdot 1)$. The constant m_f depends on the type of the dynamical system (discrete or continuous in time) and the applied tunings. For discrete systems it is usually 1. Afterwards, the mapping p, see Eq. 3.20, must be applied in order to calculate $I' = p(\mathbf{y}_i)$. This calculation takes time $t_p \in O(d)$, where d is the dimension of the phase space. Because both, m_f as well as d, are constants, it follows that the calculation of a I' is in $O(m_f + d) \subseteq O(1)$.

Remark 3.4. *For a given index I' the access of a box $M(I')$ is in $O(\log(n_s))$.*

The function $\mathbf{g}(I')$, see Eq. 3.31, can be implemented by using a hash map. Its *find(I')* operation is in $O(d \cdot \log(n_s)) \subseteq O(\log(n_s))$ whereby d is the state space dimension and also the maximal number of operations which have to be performed for determination of the order of two multi-index objects using the relation $\prec$, see Eq. 3.16.

Proposition 3.1. *The construction of the symbolic image G^s with respect to the covering C^s is in $O(n_s \log(n_s))$.*

Proof. In order to get the scan points $\mathbf{x}_i \in S(I)$ ($i = 1, \ldots, k$, k is the number of scan points), the *getBoxMapping($M(I)$)* function requires first of all the calculation of $p^{-1}(I) = \mathbf{x}_0 \in M(I)$. This takes time $t_{p^{-1}} \approx t_p \in O(1)$. If this was done, the scan points, which depend on $\mathbf{x}_0$, can be computed in $O(k \cdot d)$, see Eq. 3.27. Then for every $\mathbf{y}_i \in T(I)$ the box index I'_i with $\mathbf{y}_i \in M(I'_i)$ has to be found and the corresponding box $M(I'_i)$ must be located. According to Remarks 3.3 and 3.4 this can be done in $O(k \cdot (1 + \log(n_s)))$. So *getBoxMapping($M(I)$)* needs a total time of

$$O(1 + (k \cdot d) + k \cdot (1 + \log(n_s))) \in O(\log(n_s)). \tag{3.37}$$

Next, all cells $c_{I'} \in V(G^s)$ with $M(I') \in \tilde{C}(I)$ must be added as targets to the adjacency list of c_I. This list can also be implemented as a hash map so that the obligatory check whether $c_{I'}$ already belongs to the list is in $O(\log(e_I^{\max}))$, whereby $e_I^{\max}$ is the maximal number of target cells which is also limited to a constant by the number k of scan points per cell ($e_I^{\max} = k$). So the complexity with regard to the calculation of all target cells $c_{I'}$ for the cell c_I is in $O(k \cdot \log(k)) \subseteq O(1)$.

In order to get G^s, the target cells for all $c_I \in G^s$ must be found. So the final complexity concerning the construction of the symbolic image G^s for the covering C^s is

$$O(n_s \cdot \log(n_s)).$$

□

Remark 3.5. *The localization of the recurrent cells $RV(G^s) \subseteq V(G^s)$ using Tarjan's algorithm for calculation of strongly connected components is in $O(n_s)$.*

It is shown in Sedgewick [Sed93] that the strongly connected components can be found in linear time with Tarjan's algorithm. The extensions needed to locate recurrent cells are first the distinction, if a set γ_a, see Eq. 3.35, contains only a single recurrent cell, second the creation of the equivalence classes H_k, and third the assignment of the recurrent cells to these sets. These operations can easily be embedded into Tarjan's algorithm and do only require constant time. Therefore, the localization of recurrent cells is still in $O(n_s)$.

Remark 3.6. *The localization of $RV(G^s) \subseteq V(G^s)$ and of the shortest periodic path for each cell $c \in RV(G^s)$ is in $O(n_s^2)$.*

The analysis of Dijkstra's algorithm, see [AHU87, Tar91, CLR00], leads to the result that all shortest paths for a cell c_i can be found in time

$$\begin{aligned} t_{\text{Dijkstra}}(n_s) &\in O\left(\sum_{c_j \in G^s}\left(\sum_{c_k \in T(c_j)} O(1) + |R|\right)\right) \\ &\subseteq O(e_s + n_s \cdot |R|) \end{aligned} \tag{3.38}$$

where $T(c_j)$ is the list of target cells for c_j, R is the outer edge with the number of elements $|R| \leq n_s$ and e_s is the number of edges in G^s. As mentioned in Sec. 3.3.2, if the graph is unweighted, the outer edge R can be implemented as a queue. This improves the performance time significantly to

$$\begin{aligned} t_{\substack{\text{Dijkstra} \\ \text{unweighted}}}(n_s) &\in O\left(\sum_{c_j \in G^s}\sum_{c_k \in T(c_j)} O(1) + O(1)\right) \\ &\subseteq O(e_s + n_s \cdot 1) \end{aligned} \tag{3.39}$$

Furthermore, the number of edges per cell is limited by the number of scan points k. So the complete number e of edges in the graph G^s can not be larger than $k \cdot n_s$. Therefore we obtain

$$\begin{aligned} t_{\substack{\text{Dijkstra} \\ \text{unweighted}}}(n_s) &\in O(e_s + n_s) = O(k \cdot n_s + n_s) \\ &\subseteq O(n_s) \end{aligned} \tag{3.40}$$

The modified version of this algorithm, which locates periodic paths, does also not require more time. It is even faster because it terminates immediately whenever a shortest periodic path has been found:

$$\begin{aligned} & t_{\text{Shortest\_periodic\_path}}(n_s) \leq t_{\substack{\text{Dijkstra} \\ \text{unweighted}}}(n_s) \\ \Rightarrow\ & t_{\text{Shortest\_periodic\_path}}(n_s) \in O(n_s) \end{aligned} \tag{3.41}$$

Note that the modified Dijkstra's algorithm for calculation of shortest periodic paths needs some more checks than the original Dijkstra's algorithm for unweighted graphs. However, the performance time of these operations can be neglected for theoretical analysis.

In order to calculate the shortest path not only for one cell c_i, but for all cells in G^s, it is next necessary to start the modified Dijkstra's algorithm for each cell once. Hence

$$t_{\text{All\_periodic\_paths}}(n_s) \leq n_s \cdot t_{\text{Shortest\_periodic\_path}}(n_s). \tag{3.42}$$

Therefore the resulting time is $\in O(n_s^2)$:

$$t_{\text{All\_periodic\_paths}}(n_s) \in O(n_s^2) \tag{3.43}$$

In Remark 3.6 the worst-case scenario is considered. One should note that some more improvements of the algorithm were presented. These improvements, although not important for theoretical analysis, can lead to a significant increase of performance time for the average case. However, this depends strongly on the properties of the dynamical system in focus.

Remark 3.7. *The subdivision of a covering C^s into a covering C^{s+1} is in $O(n_s)$.*

For the subdivision of C^s, all boxes $M(I^s)$ which correspond to selected cells $c_{I^s} \in SV(G^s)$ must be subdivided. For each box $M(I^s)$ the subdivision requires m calls,

see Eq. 3.11, of a function

$$\begin{aligned} \sigma(I^s,\mathbf{j}) &= I_{\mathbf{j}}^{s+1} \\ &\quad \forall \mathbf{j} = (j_1,\ldots,j_d) \quad \forall j_k = 1 \ldots i_k^{\max} \quad \forall k = 1,\ldots,d \\ \text{with} &\quad I_j^{s+1} = \Big(j_1 + (i_1-1)\cdot i_1^{\max},\ldots, \\ &\qquad\qquad j_d + (i_d-1)\cdot i_d^{\max}\Big) \end{aligned} \tag{3.44}$$

which creates a new index $I_{\mathbf{j}}^{s+1}$, compare with Eq. 3.34. The number of these boxes, $|SV(G^s)|$, is not larger than n_s. Taking into consideration that m is a constant value and $t_\sigma \in O(1)$, it follows that

$$t_{\text{subdivide}}(n_s) \leq n_s \cdot m \cdot t_\sigma \in O(n_s \cdot k) \subseteq O(n_s) \tag{3.45}$$

Proposition 3.2. *The complete symbolic image construction process for a subdivision phase s can be done in time $O(n_s \cdot \log(n_s))$ if only recurrents cells are located, and in time $O(n_s^2)$ if the least period size for each cell should also be calculated.*

Proof. The complete construction process for the s-th discretization of the state space involves for $s > 0$ at first the subdivision of all boxes $M(I^{s-1})$ with $c_{I^{s-1}} \in RV(G^{s-1})$ into C^s. Then, for every $s \geq 0$, the construction of G^s and the localization of all recurrent cells $RV(G^s) \subseteq V(G^s)$ must be performed. According to Remarks 3.1, 3.5 and 3.7, the following performance time can be achieved:

$$\begin{aligned} t_{\text{construction}}(n_s) &\in O\Big(n_s + n_s \cdot \log(n_s) + n_s\Big) \\ &\subseteq O(n_s \cdot \log(n_s)) \end{aligned} \tag{3.46}$$

If the least period size of every cell has to be calculated then the cell location can not be done with Tarjan's algorithm. Instead, the modified Dijkstra algorithm for location of shortest periodic paths must be used. According to Remark 3.6, the performance time changes as follows:

$$\begin{aligned} t_{\substack{\text{least} \\ \text{period}}}(n_s) &\in O\left(n_s + n_s \cdot \log(n_s) + n_s^2\right) \\ &\subseteq O(n_s^2) \end{aligned} \tag{3.47}$$

□

In order to sum up the results obtained so far, one can say that, except for the calculation of the least period sizes, the time required by the discussed method

is within $O(n_s \cdot \log(n_s))$, and therefore almost ideal from the theoretical point of view, especially for large n_s. Note that these results are closely related to the proposed method for the approximation of the covering $C(I)$, Eq. 3.25. The number of edges for a cell is limited by the scan points per box, and in our case this is a constant. If another approach is chosen for the approximation of the covering like, e.g. interval arithmetic or the calculation of the Lipschitz constant [Jun00], the number of edges per cell is not longer bounded by a constant. For the results of performance analysis this would mean that most of the terms must be multiplied by n_s.

Performance analysis shows that the computation time of the algorithms is no major obstacle for the construction of the symbolic image. Instead, the crucial factor is the size of the input value n_s, i.e. the memory resources required for a computation. Note that n_s could grow almost exponential during the subdivision process. The size and growth rate of n_s depend hereby not only on the investigation task, i.e. the dimension of those objects which are the subjects of investigation, but also on the specific properties of the focused dynamical system. Practical application has shown that a high growth rate of n_s is a limitation for many computations. Hence, it should be the main concern to keep the number of cells n_s low and avoid an high growth rate during subdivision. Often this can be achieved by appropriate parameter settings or tunings of the method, see Section 4.2.

To give a rough overview about the computation times necessary for specific investigations, we present in the following some reference times. The used reference machine for all these calculations was an `Asus L3000D` laptop with an `AMD Athlon XP-M 1400+` processor and `512MB SDRAM`.

3.5 Accuracy of the Computations

Let us next consider the accuracy of the numerical calculation. One should recall that we do not calculate specific points in the domain space but boxes $M(I)$ with some extent. These boxes are always only an outer covering of a solution. In our implementation, a box $M(I)$ is defined as a uniform grid box. The size of such a grid box defines the accuracy ε of the calculation. Let us denote by d_k the edge length of a grid box $M(I)$ on the dimension axis k, and by L^s the union of boxes which correspond to the selected cells $SV(G^s)$ in the symbolic image G^s constructed after the s-th subdivision. Then these boxes in L^s are neighborhoods (or an outer covering) of a solution S. The basic principle of symbolic analysis is that the sequence of embedded neighborhoods $L^0 \supseteq L^1 \supseteq \cdots \supseteq L^m$ gets for every

subdivision step $s = 0, \ldots, m$ closer to S in the way that, if the largest edge length tends to zero as s becomes infinite, then,

$$\lim_{s \to \infty} L^s = \bigcap_s L^s = S. \tag{3.48}$$

See also Theorems 3.1 and Theorems 3.2 as examples of this principle. Unfortunately, for practical numerical calculations there is a minimal edge length which limits the accuracy. Reason for this is that the n-dimensional state space covered by M, see Eq. 3.10, gets divided into regions which are identified by multi-indices $I \in \mathbb{N}^n$, Eq. 3.13. As mentioned above, a value i_k of the k-th component of the multi-index I is represented by an integer value, and for every computation machine there exists a constant N_{int} giving the largest number which can be represented as an integer value. Consequently, we have the limitation $i_k^{\max} \leq N_{\text{int}}$. Therefore, every edge length d_k is limited to

$$d_k^{\min} = \frac{M_k^{\max} - M_k^{\min}}{N_{\text{int}}} \tag{3.49}$$

which means that the minimal error ε_k can only shrink down to $\varepsilon_k \geq d_k^{\min}$.

Note that this limit is not specific for the presented method but only for the implementation presented in this work. Furthermore, it is possible to extend this limit to any size by taking a different representation of a number i_k, though this implies a higher memory consumption.

Using our reference machine mentioned above, symbolic images of a size up to $\approx$ 2 000 000 cells can be constructed. Furthermore, the largest number representable by the used hardware architecture is $N_{\text{int}} = 2^{32}$, so that we obtain

$$\varepsilon_k \geq d_k^{\min} \approx 10^{-9} \cdot (M_k^{\max} - M_k^{\min}). \tag{3.50}$$

This is the limit for the accuracy of every calculation on our reference machine.

3.6 Numerical Case Studies

In order to demonstrate the capabilities of symbolic analysis we present some typical examples of global analysis. The main aim hereby is to demonstrate what kind of results can be obtained with the basic investigation techniques presented in Sec. 3.3, and how the parameters of the method have to be adjusted for specific investigation tasks.

3.6.1 Ikeda Map

We start with a 2-dimensional map, namely the Ikeda map [Ike79]. The system is defined as

$$\begin{aligned} &\mathbf{x}(n+1) = \mathbf{f}_I(\mathbf{x}(n)), \\ &\mathbf{f}_I : \mathbb{R}^2 \to \mathbb{R}^2, \quad \mathbf{x} = (x,y)^T \\ &\mathbf{f}_I(\mathbf{x}) = \begin{pmatrix} r + a\,(x\cos g(x,y) - y\sin g(x,y)) \\ b\,(x\sin g(x,y) + y\cos g(x,y)) \end{pmatrix} \\ &\text{with} \quad g(x,y) = c_1 - c_2/(1+x^2+y^2) \end{aligned} \tag{3.51}$$

and occurs in the modeling of optical recording media. The Ikeda map is interesting for symbolic image analysis because it contains stable and unstable periodic points as well as a chaotic attractor which we tried to detect.

A comprehensive study of the system by symbolic image methods can be found in [Osi04]. However, in that work only the results are presented. No details about the numerical calculation are mentioned. For this reason, some of the computations are discussed here again. The numerical simulations have been carried out for the parameter values $a = b = 0.9$, $c_1 = 0.4$, $c_2 = 6.0$ and $r = 0.9$. According to other numerical results obtained up to date, see [MY00] and references therein, exists at these parameter values a chaotic attractor $\mathcal{A}$, two unstable fixed points $P_{1,2}$ (saddle points) and the stable fixed point P_3:

$$\begin{aligned} P_1 &= \begin{pmatrix} 0.4819 \\ 0.2545 \end{pmatrix}, \\ P_2 &= \begin{pmatrix} 1.1987 \\ -2.3769 \end{pmatrix}, \\ P_3 &= \begin{pmatrix} 3.0027 \\ 3.8945 \end{pmatrix} \end{aligned}$$

Additionally, there exists the unstable 2-periodic orbit T_1^2, as well as two unstable 3-periodic orbits $T_{1,2}^3$:

$$\begin{aligned} T_1^2 &= \left\{ \begin{pmatrix} 0.5964 \\ 0.6394 \end{pmatrix}, \begin{pmatrix} 0.4497 \\ -0.6453 \end{pmatrix} \right\} \\ T_1^3 &= \left\{ \begin{pmatrix} 0.8091 \\ 0.7834 \end{pmatrix}, \begin{pmatrix} 0.9960 \\ -1.0090 \end{pmatrix}, \begin{pmatrix} -0.0280 \\ -0.8758 \end{pmatrix} \right\} \\ T_2^3 &= \left\{ \begin{pmatrix} 1.3512 \\ -0.0707 \end{pmatrix}, \begin{pmatrix} 0.6568 \\ -1.1932 \end{pmatrix}, \begin{pmatrix} -0.2418 \\ -0.4462 \end{pmatrix} \right\} \end{aligned}$$

Note that T_1^2 and $T_{1,2}^3$ lie close to the chaotic attractor $\mathcal{A}$.

We start the global analysis of the system by localization of the chain recurrent set. As already mentioned, the chain recurrent set contains all kind of return trajectories and, hence, should give an overview about the areas of interest for further investigation. We set the area of investigation in the domain space to $M = [-5.0;5.0] \times [-5.0;5.0]$. This area M is initially divided into a covering C^0 of 20×20 boxes. Then the symbolic image G^0 is constructed for C^0. We apply the Tarjan algorithm, see Sec. 3.3.1, to detect the recurrent cells. The construction and cell detection process was repeated for 4 subdivisions. In each step the boxes are subdivided into 4×4 new smaller boxes. The results of such a calculation can be seen in Fig. 3.4. After four subdivision steps, the distinct features of the Ikeda mapping can be found in the different recurrent sets of the symbolic image. Three areas in the state space are detected. One of them represents the stable point P_3, the other one the unstable saddle point P_2, and the last one contains all cells representing the chaotic attractor $\mathcal{A}$. It is worth mentioning that chaotic attractors can be found by computation of the chain recurrent set because their skeleton is typically build up from unstable periodic cycles [Cvi91, Cvi92] and, therefore, recurrent points. The calculation of the symbolic image takes less than 2 minutes. In Tab. 3.1 the number of cells in the symbolic image and the number of located recurrent cells for every subdivision level are shown. We observe that the number of recurrent cells grows during the subdivision process by factor ≈ 10. This high growth rate is due to the fact that it depends on the dimension of those objects which are the subject of localization, see also [Jun99, Mis02]. In this case, one of the subjects of localization is the chaotic attractor $\mathcal{A}$ which has a dimension close to 2. Note also that the periodic orbits T_1^2 and $T_{1,2}^3$ as well as the fixed point P_1 lie inside the computed outer covering of the chain recurrent set. Due to the fact that they are close to the attractor $\mathcal{A}$, they can not be distinguished from it by this kind of computation.

Our next target is the localization of the periodic orbits T_1^2 and $T_{1,2}^3$. This requires, of course, the usage of the time consuming variant of the cell location algorithm based on the Dijkstra algorithm, see Sec. 3.3.2. We took the same area of investigation, $M = [-5.0;5.0] \times [-5.0;5.0]$, than for the last computation but select only the cells with period size $p' < p = 3$ for further subdivisions. After an initial subdivision of M into 20×20 boxes, the boxes get subdivided into 8×8 new smaller boxes in every subdivision. The construction and cell detection process was repeated for 9 subdivisions. The results of the calculation in the area $[-1.5;2.5] \times [-1.5;2.0]$ can be seen in Fig. 3.5(a), namely the points belonging to P_1, T_1^2 and $T_{1,2}^3$. Note that also the points $P_{2,3}$ were detected but no other orbits with a period ≤ 3. Here it turns out that the usage of a sufficient number of

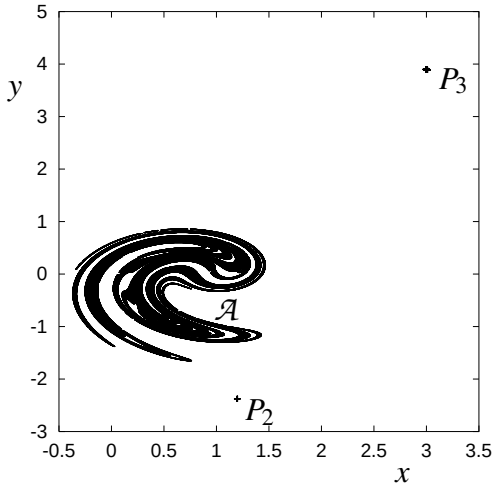


Figure 3.4: *Ikeda system: Numerical computation of an outer covering of the chain recurrent set. It contains the chaotic attractor $\mathcal{A}$ as well as the fixed points $P_{2,3}$.*

scan points is important. In this example, every box contained 8×8 scan points scattered over the box, four more points close to the box corners and another one in the center. Such a high number is needed to acquire all the periodic orbits. If less scan points are chosen, another parameter must be set for error tolerance, see Sec. 4.1.2, or some of the periodic cycles will not be detected. Although this calculation uses the more time consuming period detection algorithm, the computation takes less than 30 seconds on the reference machine. This is due to the fact that only very few cells have a period size smaller or equal than 3. In our calculations, not more than 97 cells per subdivision step fit to this criterion. So the size of the symbolic images can be kept very small. However, one should notice that the performance time can increase exponentially if the parameter p is set to a higher value and more such cycles with $p' \leq p$ exist.

A serious problem we come across in this computation is that for most points not only one box corresponding to a periodic cell is found, but several boxes in the neighborhood. In this case we get up to 5 boxes as an outer covering for each periodic point instead of one box. In the following, we will refer to this phenomenon as *clustering*. Empirically, it has shown that clustering can be considered as one of the most common and crucial numerical artifacts occurring in the context of symbolic analysis. Due to clustering, the growth rate of the number of cells in a symbolic image increases during the subdivision process, and the accuracy of the computation shrinks. Moreover, the analysis of computed data is more difficult.

Subdivision level s	Phase space discretization	$\lvert V(G^s)\rvert$	$\lvert RV(G^s)\rvert$
0	20×20	400	79
1	80×80	1264	415
2	320×320	6640	3018
3	1280×1280	48288	27878
4	5120×5120	446048	284727

Table 3.1: *Ikeda system: Computation of the recurrent cells. The number s marks the level of subdivision. The phase space discretization of the area $M = [-5.0;5.0] \times [-5.0;5.0]$ is shown. Furthermore, the number of cells belonging to a symbolic image, $|V(G^s)|$, and the number of localized recurrent cells $|RV(G^s)|$.*

Theoretically, the following accuracy, compare Sec. 3.5, could be achieved for the calculated points:

$$\varepsilon \leq \frac{1}{20} \cdot \frac{1}{8^9} \cdot 4 \approx 2 \cdot 10^{-9}.$$

However, in practice, the error is higher because of clustering. If taking this into account and analyzing the computed results, the error increases to $\varepsilon \leq 1 \cdot 10^{-7}$.

Because $\mathcal{A}$ is a chaotic attractor, one can expect to find in its vicinity some unstable limit cycles with periods higher than 3. So first we increased p to 6 and then to 14. Some results of these calculations are presented in Fig. 3.5(b), which shows two of the detected unstable 5- and 6-periodic orbits, and Fig. 3.5(c), an overview of all detected 6- and 13-periodic points. Remarkably, the symbolic images for $p = 6$ contained not more than 325 cells, for which the corresponding boxes got subdivided and thus the calculation did not take much more computation time than in the first case (≈ 30 seconds). But the location of cells with a period size ≤ 14 consisted of up to 27000 selected cells. Boxes corresponding to each of them get subdivided into 8×8 new smaller boxes, so that the symbolic images had up to 1700000 cells. Therefore, the calculation took around eight hours in this case.

Until now we investigated the Ikeda system for fixed parameter values, as described above. Using the methods of symbolic analysis under variation of some parameters, interesting results can be obtained as well. For instance, one can observe the bifurcations which causes the emergence of unstable periodic orbits. These periodic orbits determine the structure of the chaotic attractor discussed above. Performing this task, we consider the area $M = [-0.4;1.5] \times [-1.7;1]$ in

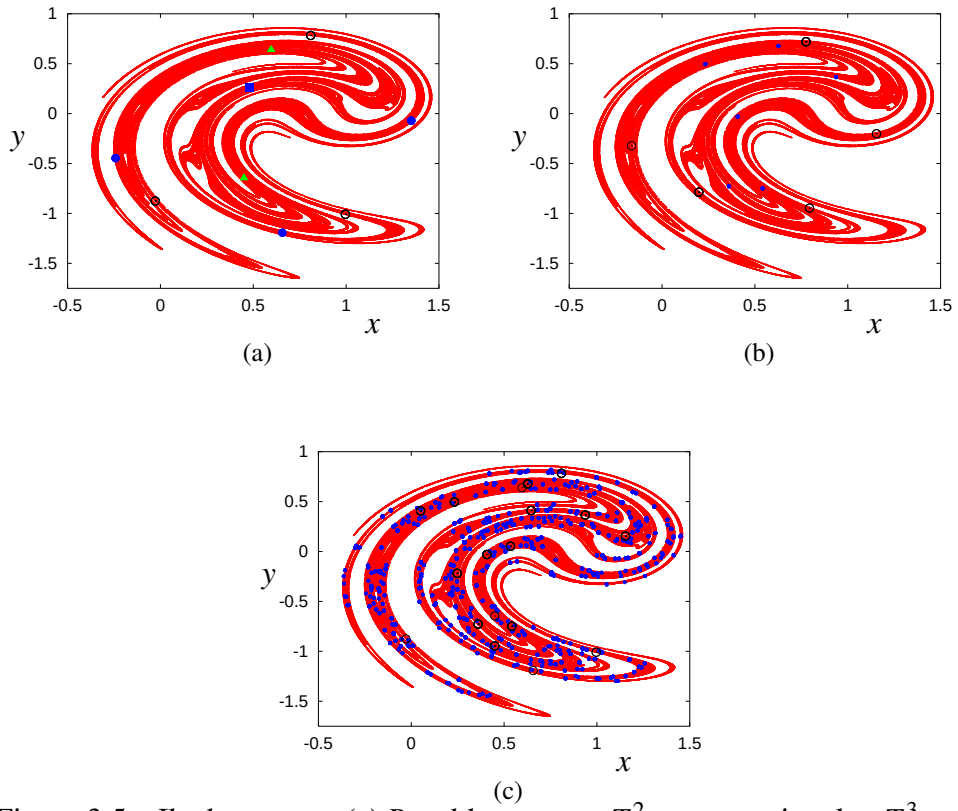
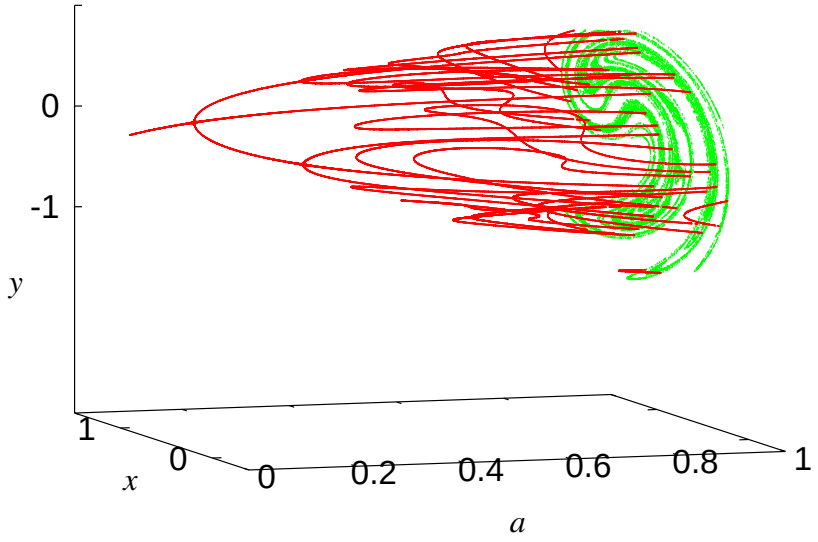


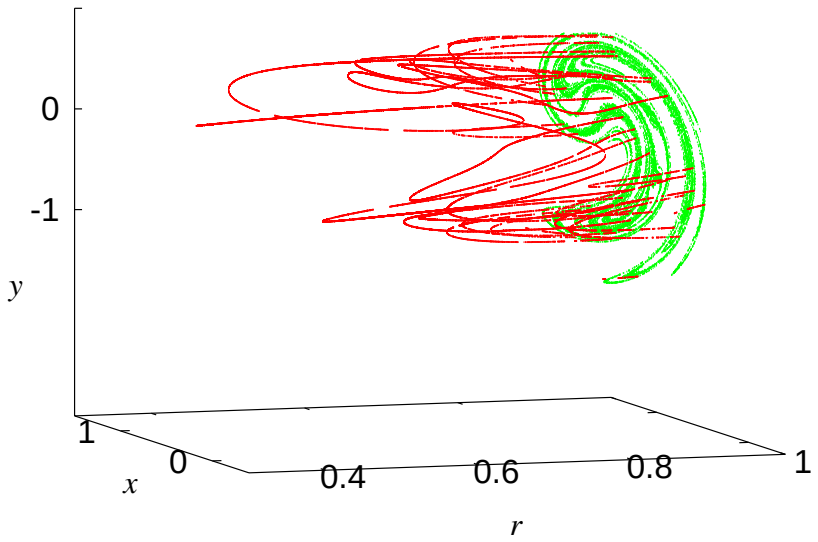
Figure 3.5: *Ikeda system: (a) P_1 – blue square, T_1^2 – green triangles, T_1^3 – empty circles, T_2^3 – blue circles. (b) Some detected unstable limit cycles with periods* 5 *(empty circles) and* 6 *(blue circles). (c) All detected unstable* 6*-periodic (empty circles) and* 13*-periodic (blue circles) points. Note that the chaotic attractor $\mathcal{A}$ in the background is visible for better orientation but was not calculated by the same computation.*

the state space and calculate the periodic orbits up to period six. Using an initial subdivision into 20×20 boxes and performing 4 subdivision steps, whereby each box is divided into $2 \times 8 \times 8$ smaller boxes, we obtain the results shown in Fig. 3.6. The parameters a and r are varied in the interval $[0;0.9]$. The other parameters are kept fixed to the same values as above. In both experiments we observe a period doubling bifurcation scenario and a large number of saddle-node bifurcations.

In this section we have only applied basic investigations on the Ikeda system. It turned out that one could already gather a large amount of information regarding the global analysis. However, the Ikeda system will be again subject of discussion in Sec. 5.7.2. Then, advanced investigation methods of symbolic analysis are applied on it.



(a)



(b)

Figure 3.6: *Ikeda system: Periodical points up to period 6 under variation of parameters a and r.*

3.6.2 Coupled Logistic Map

The preceding example was chosen because the results of the investigation could easily be reproduced and verified by the use of other methods like forward iteration. In order to illustrate the capabilities of symbolic analyis, we give now an example of an investigation which can not be done by any other method known to the authors. Therefore, we take a look at another 2-dimensional map, the coupled logistic map defined by:

$$\begin{aligned} &\mathbf{x}(n+1) = \mathbf{f}_C(\mathbf{x}(n)), \\ &\mathbf{f}_C : \mathbb{R}^2 \to \mathbb{R}^2, \quad \mathbf{x} = (x,y)^T \\ &\mathbf{f}_C(\mathbf{x}) = \begin{pmatrix} (1-r)\, g(x,a) + r\, g(y,b) \\ r\, g(x,a) + (1-r)\, g(y,b) \end{pmatrix} \\ &\text{with} \quad g(x,m) = m\, x\, (1-x). \end{aligned} \tag{3.52}$$

The system, as presented here, can be considered as a 2-dimensional case study of coupled map lattices [Kan93] for the logistic map [GSE93]. In our context, the study of this system reveals some interesting dynamics. As for the Ikeda system, we computed the chain recurrent set and periodic orbits. We did not only find an attractor but also fractal structures which, in contrast to the attractor, can not be revealed by forward iterations.

For all our investigations, we fixed the parameter settings to $a = b = 3.8$ and $r = 0.07$. Analytically, it is easy to show that, due to $a = b$, we have symmetric behavior with respect to the diagonal $y = x$. This means that orbits become symmetric if one interchanges the x- and y-coordinates, and that all points on the diagonal at $y = x$ form an invariant set $\mathcal{D}$. By numerical analysis based on forward iterations and calculation of Lyapunov exponents, one can find out, that the system is governed by a single attractor $\mathcal{A}$ which consists of two symmetric parts in the phase space, see Fig. 3.7.

Our first investigation of the system by symbolic image analysis was the computation of the chain recurrent set. We initially divided the area $M = [0.0;1.0] \times [0.0;1.0]$ into 5×5 boxes. In each subdivision step, a box gets divided into 3×3 new ones. After 5 subdivisions the outer covering of the chain recurrent set consists of 430 000 boxes with a side length $\approx 1 \cdot 10^{-3}$. It is important to mention that a high number of scan points is required. If taken too little, large parts of the chain recurrent set get lost during the first subdivisions. Hence, for our investigation we covered each box with a regular grid of 100 scan

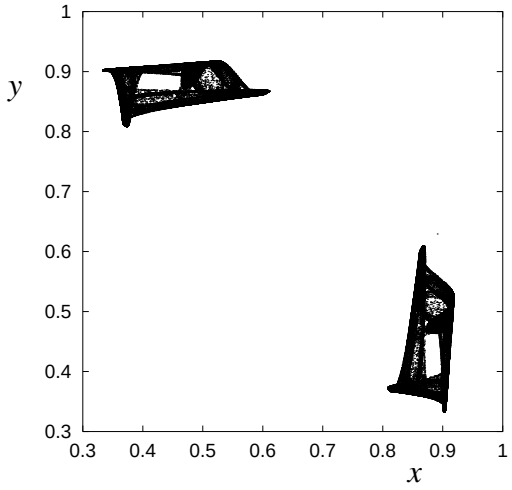


Figure 3.7: *Coupled logistic map: Numerical approximation of the attractor $\mathcal{A}$.*

points. Applying these settings, the computation takes around 8 minutes, and its results can be seen on Fig. 3.8(a). The chain recurrent set does not only consist of the chaotic attractor but also of fractal structures which are symmetric with respect to the diagonal. Note that these fractal structures are unstable entities. Orbits started in a neighborhood of the chain recurrent set are attracted by the attractor $\mathcal{A}$. We observed that even orbits started in the area covered by the computed fractal structures are attracted by $\mathcal{A}$. However, this can be explained by the fact that our numerical computation produced an outer covering of the real chain recurrent set and, hence, covers also the chain recurrent set's neighborhood. To the authors' knowledge there is no other method of numerical investigation, except the related set-oriented approach by Dellnitz *et.al.* [DFJ01], which is capable to reveal these structures. In Fig. 3.8(b) we colored each component of the chain recurrent set differently. It is clearly to see that there are 4 distinct equivalent recurrent sets, one of them represents $\mathcal{A}$, and another one a 2-periodic unstable orbit in the holes of $\mathcal{A}$.

In order to verify our results, we also computed periodic orbits. We used the cell location algorithm based on the Dijkstra algorithm, see Sec. 3.3.2, and computed all periodic points with a periodicity ≤ 8. We applied 17 subdivisions so that the error $\varepsilon \leq 1 \cdot 10^{-8}$. The computation took around 25 minutes, and we got 614 periodic points, see Figs. 3.8(c) and 3.8(d). It can be observed that periodic orbits are scattered over the whole area designated by the approximation of the chain recurrent set. We confirmed our results by applying a Quasi-Newton method to locate periodic orbits as proposed in [NY97], see also Sec. 2.2. This method

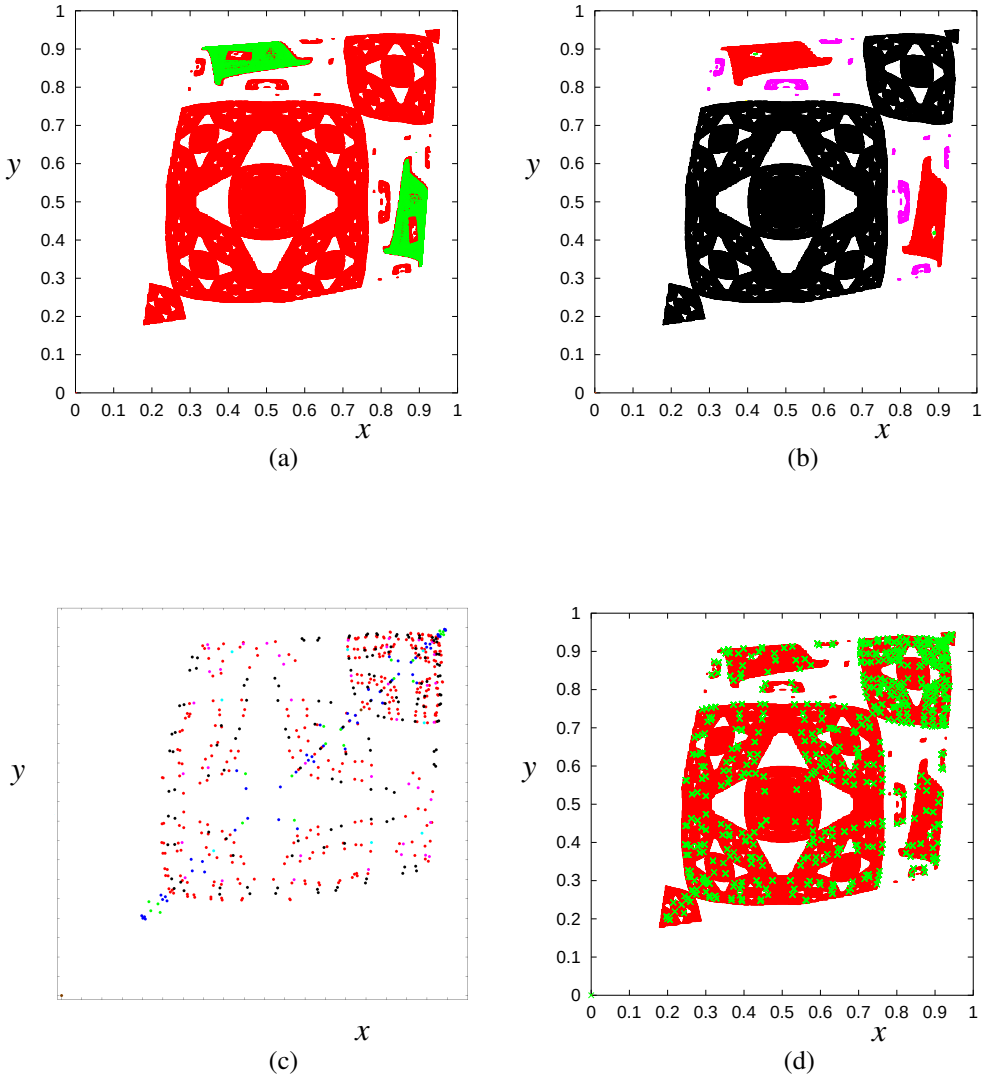


Figure 3.8: *Coupled logistic map: (a) Numerical approximation of the chain recurrent set (red) and the attractor $\mathcal{A}$ (green). (b) The areas covered by different components of the chain recurrent set are shown in different colors. (c) Detected periodic points. Periodicity: 1 (brown), 2 (cyan), 4 (magenta), 5 (green), 6 (black), 7 (blue) and 8 (red). (d) Detected periodic points and the chain recurrent set*

solves the equation

$$\mathbf{f}^{[p]}(\mathbf{x}) - \mathbf{x} = 0,$$

whereby p is the periodicity of the orbit. As start values for the Newton iteration we took our computed periodic points. For each of them we found a periodic orbit of the same period in the immediate neighborhood, giving evidence for the correctness of our calculations. Furthermore, we also checked that each periodic orbit is unstable.

Combining the results of our numerical computations so far, we find strong evidence for the hypothesis that the computed fractal structure of the chain recurrent set is an outer covering of a set of unstable periodic orbits of any size. This reminds us of the hypothesis of Cvitanović [Cvi91] regarding periodic orbits as the skeleton of chaotic attractors. However, the fractal structure we observe here is not an attractor.

Chapter 4

Extensions and Tunings

In the last chapter we have introduced an implementation for the basic framework of symbolic analysis. It is now our intention to focus on several important aspects of a practical application. For this purpose we propose some extensions and tunings of the original concepts. We discuss these techniques and, if necessary, also mention some of their implementation aspects and useful heuristics for an efficient usage. Furthermore, several numerical case studies demonstrate their implications in practice.

The extensions mentioned here regard the construction of the symbolic image graph. We consider two important extensions. One is the integration of dynamical systems continuous in time, the other a technique for the better approximation of the image of a box. Both of these extensions aim to improve the construction of the graph and are in accordance with the theoretical concepts. This is not the case for the tunings. The tunings do not aim on providing a more precise symbolic image. Rather more, the target is to introduce techniques which allow the application of our investigation methods in scenarios where the construction of a regular symbolic image graph is not or only to a limited extent possible. Such a case happens, for instance, if the growth rate of the cells in the symbolic images is very high during subdivision so that the memory resources get exceeded and the desired investigation can not be finished. The essence of our approaches for tuning is that some aspects important in theory are neglected for the sake of a successful practical application of the investigation method. The tunings proposed in this chapter are motivated by empirical studies of our computations. Their application often leads to a more efficient and/or more precise calculation.

4.1 Extensions for the Graph Construction

We introduce additional techniques for the construction of the symbolic image graph. The intention is to improve and extend the construction process described in the last chapter.

4.1.1 Dynamical Systems Continuous in Time

Only dynamical systems discrete in time have been discussed so far. The symbolic image for such a system

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k),\ x_k \in \mathcal{M}$$

can be constructed by performing one iteration which means simply applying the system function $\mathbf{f}(\mathbf{x})$ on the points $\mathbf{x} \in M(I)$ lying in a box $M(I)$ of a certain covering, see Eq. 3.28. If we are dealing now with systems continuous in time given by an ODE, i.e. $\dot{\mathbf{x}} = \mathbf{F}(t,\mathbf{x})$, some kind of mapping is required which transforms an orbit continuous in time into one discrete in time. As already mentioned in Sec. 2.1, a shift operator along trajectories is needed. Such a shift operator $\phi(t,t_0,\mathbf{x}_0)$ is considered to be the solution of the vector field $\mathbf{F}$ with an initial condition $\phi(t_0,t_0,\mathbf{x}_0) = \mathbf{x}_0$.

Several approaches exist toward this task. Ideally, a shift operator for a Poincaré mapping can be obtained analytically. This means that an explicit shift operator ϕ can be found so that a Poincaré map $\mathbf{f}(\mathbf{x}) = \phi(\omega,0,\mathbf{x})$ can be constructed for a fixed period ω. Unfortunately, this is usually not possible. Alternatively, in [Mis02] the use of local Poincaré sections is proposed. The dynamics of multiple $(n-1)$-dimensional hypersurfaces are studied which are transversal to parts of the system's flow. An advantage of both approaches is that the dimension of the investigated system is reduced by one. On the other hand, their application is limited.

In our implementation, we use a more general method – a stroboscopic mapping with a fixed discretization time t. This approach is always applicable if the underlying differential equation is autonomous, i.e. the vector field $\mathbf{F}$ does not depend on t. Then the shift operator has the form $\mathbf{f}(\mathbf{x}) = \phi(t,\mathbf{x})$ with $\phi(0,\mathbf{x}) = \mathbf{x}$. It can be calculated by solving the equation

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{x}(t)) \tag{4.1}$$

for the time t and initial conditions $\mathbf{x}(0) = \mathbf{x}$. We assume a fixed $t > 0$. In that case, $\phi(t,\mathbf{x})$ is also called a *time-t map*. Such a time-t map is a restriction of ϕ to $\mathcal{M} \times t\mathbb{Z}$ and, hence, a discretization of the dynamical system continuous in time.

Note that similar approaches for a discretization were also proposed by [Jun99] and [Pil99].

Consider now that in the context of a computer implementation it is suitable to use a small integration step size Δt for the applied integration method in order to minimize numerical errors. Hence, we do not calculate $\phi(t,\mathbf{x})$ explicitly. Instead, we use an integration step size $\Delta t = t/n$ with $n \in \mathbb{N}$ and iterate ϕ for n times so that

$$\phi(t,\mathbf{x}) = \phi(\Delta t \cdot n,\mathbf{x}) = \phi^n(\Delta t,\mathbf{x}).$$

This approach allows the numerical computation of time-t maps for any precision, independently of the chosen discretization time t. In the following, we use the notation $\mathbf{f}(\mathbf{x}) = \phi(\Delta t,\mathbf{x})$ so that the time-t map for a $t = \Delta t \cdot n$ is given by $\mathbf{f}^{[n]}(\mathbf{x})$. Hence, the symbolic image is constructed assuming $\mathbf{f}^{[n]}(\mathbf{x})$ as the system function instead of $\mathbf{f}(\mathbf{x})$, see also the comments to Eq. 3.28.

Both values, Δt and n, are user-defined parameters. Suitable settings depend on the properties of the investigated dynamical system as well as on the characteristics of the particular symbolic image construction. Hereby, the choice of an appropriate integration step size Δt should mainly depend on the properties of the investigated dynamical system and the used integration method. The aim is to keep the numerical error reasonably small. On the other hand, a suitable setting for the number of integration steps n should be chosen with respect to the properties of the symbolic image, especially the used box size, as well as the velocity of the dynamical system.

It turns out that an appropriate setting of Δt and n is a nontrivial task upon which the accomplishment of the symbolic image construction highly depends. This is mainly due to the fact that a continuous trajectory does not jump from a point $\mathbf{x}_n$ to $\mathbf{x}_{n+1}$ in the domain space $\mathcal{M}$ but rather more either moves from the area covered by a box $M(I)$ to the area covered by a neighboring box $M(I')$, or it stays within the same box. Hence, we distinguish two critical cases:

1. If t is chosen too small, some trajectories might not leave their boxes $M(I)$ in our simulation although they would do so at a later time $t' > t$. Then a cell c_I corresponding to such a box $M(I)$ might appear to be 1-periodic although $M(i)$ contains only transient, i.e. non-recurrent, dynamics. Note that this behavior can also happen for systems discrete in time. However, time-t maps are usually much stronger affected.

2. If t is chosen too large, the simulation of a trajectory started in a box $M(I)$ might not stop after this trajectory has entered the next neighboring box

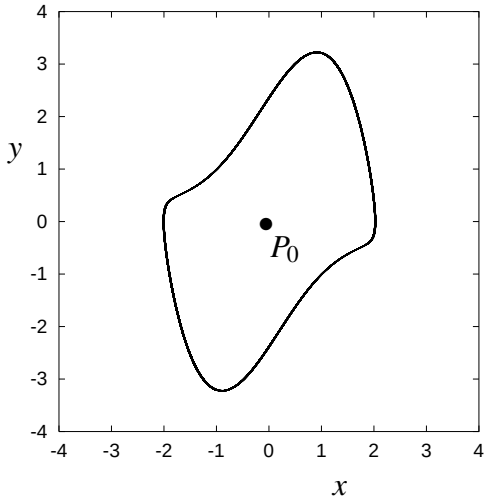


Figure 4.1: *Van der Pol system: A numerical approximation of the chain recurrent set. It consists of a stable limit cycle and the fixed point P_0.*

> $M(J)$ but only after it has entered a later box $M(K)$ on the path of the trajectory. Then edges between some cells c_I and c_J are not detected and spared out. This might have the effect that the symbolic image graph is not an appropriate representation of the system's flow, especially if the velocity of the flow strongly fluctuates. As a result, for instance, some cycles might not be, or only partly, located.

As can be seen, the use of a time-t map requires the consideration of problems which do not exist or are less important in the context of ordinary mappings. Some of the tunings presented later are also concerned with that and propose ways to better control the above mentioned difficulties.

In order to demonstrate the construction of a symbolic image graph for dynamical systems continuous in time, let us consider the Van der Pol system, defined by

$$\begin{aligned} &\dot{\mathbf{x}}(t) = \mathbf{F}_{VdP}(\mathbf{x}(t)), \quad \mathbf{F}_{VdP} : \mathbb{R}^2 \to \mathbb{R}^2, \quad \mathbf{x} = (x,y)^T \\ &\mathbf{F}_{VdP}(\mathbf{x}) = \begin{pmatrix} y \\ \gamma\, y\, (1-x^2) - x \end{pmatrix} \end{aligned} \tag{4.2}$$

Investigation results for this 2-dimensional dynamical system continuous in time can be found, for instance, in [AFH94]. For $\gamma = 1.5$ the system possesses an unstable fixed point $P_0 = (0,0)^T$ and a stable limit cycle around this point.

We try to reproduce these results. As mentioned in Sec. 2.3, limit cycles of a system continuous in time can be approximated by computation of the chain recurrent set. The symbolic image is constructed for the state space area $M = [-3.5;3.5] \times [-3.5;3.5]$. An initial subdivision of M into 50×50 boxes and further divisions of boxes into 4×4 new ones are used. The trajectories are approximated by setting the integration step size $\Delta t = 0.001$ and the number of integration steps $n = 100$. In order to compute the integration step $\phi(\Delta t, \mathbf{x})$, the Runge-Kutta method was applied. The construction for four subdivisions requires a computation time of around 15 minutes and produces images of up to 800 000 cells. The results can be seen in Fig. 4.1. The stable limit cycle as well as the unstable fixed point P_0 were found with an error of $\varepsilon \leq 10^{-2}$ after M was divided into $12\,800 \times 12\,800$ boxes. Clustering can be observed for the performed calculation. P_0 is not represented by a single cell but rather more by a bundle of 426 distinct recurrent cell sets, each containing only one periodic cell, which corresponds to a box in the neighborhood of the fixed point. Furthermore, the outer covering of the stable limit cycle is represented in the symbolic image graph by a single recurrent cell set, containing 202 886 cells.

4.1.2 Error Tolerance for Box Images

As already stated, the construction of symbolic images requires the approximation $\tilde{C}(I)$ of the covering $C(I)$, see Eqs. 3.24 and 3.25. Note that for our proposed implementation there is $\tilde{C}(I) \subseteq C(I)$, and so some boxes may get lost. This behavior can be reduced by usage of a large number of scan points for each box. A disadvantage of this approach is that the computation time may become inappropriately large. As another solution of the problem one can extend the covering $\tilde{C}(I)$ by boxes which correspond to boxes in its neighborhood. We define a small constant ε and introduce the extended covering

$$C^{\text{ext}}(I) = \big\{ M(I') \quad | \quad \begin{array}{l} \exists M(J) \in \tilde{C}(I), \\ \exists \mathbf{x} \in M(J), \exists \mathbf{y} \in M(I'), \\ \rho(\mathbf{x},\mathbf{y}) \leq \varepsilon \big\}, \end{array} \tag{4.3}$$

Note that a suitable setting of the constant ε depends on the edge length $d_k(M(I))$ of the boxes, Eq. 3.12, and hence on the subdivision level. In our implementation, the user can define a parameter e, which is in the following denoted as the error tolerance, so that

$$\varepsilon_k = e \cdot d_k \tag{4.4}$$

for the k-th phase space component. The condition $|x_k - y_k| \leq \varepsilon_k$ is used instead of $\rho(\mathbf{x},\mathbf{y}) \leq \varepsilon$ in Eq. 4.3. Note that d_k is the generalized edge length of the boxes

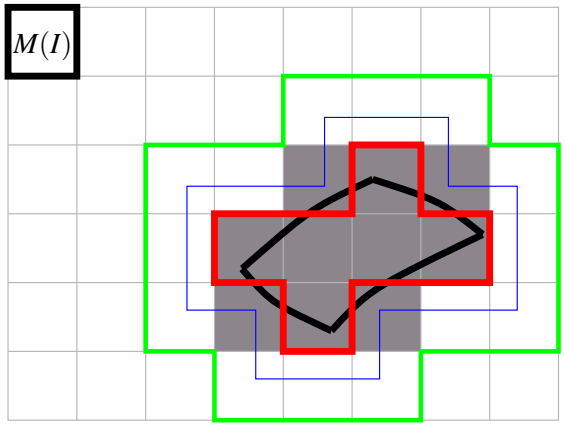


Figure 4.2: Influence of the error tolerance parameter.

in a covering C.

Example 4.1. *The influence of this error tolerance parameter is illustrated in Fig. 4.2. The shape in the middle part of the picture represents the image of the box $M(I)$. The gray area is the theoretical covering of the image $C(I)$. As one can see, some boxes of this covering may be lost in the covering $\tilde{C}(I)$ obtained by numerical calculation (shown red). In order to avoid this, the numerically obtained covering is extended by the area marked by the blue line. Then the resulting covering $C^{ext}(I)$ (shown green) contains the complete theoretical one.*

In practice, this parameter was used to detect p-periodic trajectories if they can not be found otherwise, whereby a setting of $e = 0.1$ proved to be sufficient in our simulations. One should keep in mind that the use of this parameter usually increases the size of the symbolic image and, therefore, it should only be applied if necessary. Furthermore, it is often a good alternative to increase the number of scan points $S(I)$ for a more precise calculation instead of applying error tolerance. Some of the scan points should then be placed close to the corners of the boxes.

Note that this approach was motivated by the technique described in [Jun00]. If ε is calculated by Lipschitz constants as proposed by Junge, it could be guaranteed that $C(I) \subseteq C^{\text{ext}}(I)$.

4.2 Tunings for the Graph Investigation

The tunings which are presented in this section are motivated by the results of our empirical experience. It has shown that one of the most crucial limitations of investigations based on symbolic analysis is due to a high growth rate of the number of cells during subdivision. This growth rate can be exponential but should depend on the dimension size of the localized objects. However, due to the complexity of the dynamics, this is often not the case. Instead, we observed in many cases that much more cells are selected for subdivision than those covering the solution. The phenomenon of clustering, which we already mentioned earlier, see Sec. 3.6.1, is an example of such behavior. Taking the theoretical point of view, the selection of too many cells does not matter. By successive application of the subdivision process, the solution will eventually be detected. However, taking the practical point of view, one has to deal with limited resources. That means that the number of applicable subdivisions is limited. Firstly, by the memory space of the computation machine, which only allows the storage of a symbolic image graph of a limited size. Secondly, by the fact that the division of the phase space is limited by a constant N_{int}, see Sec. 3.5.

For the above mentioned reasons, it is now our strong concern to avoid the selection of cells for subdivision which do not contain a solution. This aim can only be achieved by a change of paradigm. The target is not anymore the rigorous construction of the symbolic image graph for a phase space discretization. We are not interested in providing all existing edges between the cells as requested in the theoretical approach, but rather more only those edges which are necessary for the detection of the solution. The aim of the proposed tunings is to approach this goal. By doing so, we are also aware of the fact that some important information might get lost. However, empirical studies have shown that numerical investigations are mostly limited by performance resources instead of an insufficient approximation of the symbolic image. A significant reason for this is that the method is typically quite robust. Note that also the decision to approximate the image of a cell only by scan points instead of a rigorous covering, see Sec. 3.2.2, was motivated to a large extend by these considerations.

4.2.1 Use of Higher Iterated Functions

When dealing with dynamical systems discrete in time $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n)$, the points $\mathbf{y} \in \tilde{T}(I)$, which represent the images of $\mathbf{x}$, are calculated as direct successors of the scan points: $\mathbf{y} = \mathbf{f}(\mathbf{x})$. However, in some cases it is more suitable to use an

iterated function of $\mathbf{f}$ and calculate the image points by

$$\mathbf{y} = \mathbf{f}^{[n]}(\mathbf{x}), \quad n > 1 \tag{4.5}$$

In other words, the symbolic image is not constructed for the function $\mathbf{f}$ but for the n-th iterated function $\mathbf{f}^{[n]}$. In the following, we will denote a symbolic image constructed for $\mathbf{f}$ by G_f and for $\mathbf{f}^{[n]}$ by $G_{f^{[n]}}$.

Obviously, the symbolic image graph $G_{f^{[n]}}$ with $n > 1$ differs from G_f. More precisely, $G_{f^{[n]}}$ might have less edges than G_f. However, $G_{f^{[n]}}$ is still useful for investigations. In order to clarify this, we introduce some theorems about the relations of $\mathbf{f}^{[n]}$ and $\mathbf{f}$ with regard to invariant sets.

Proposition 4.1. *If $Q \subset M$ is an invariant set for $\mathbf{f}$, then also for any $\mathbf{f}^{[n]}$, $n \in \mathbb{N}$, i. e.*

$$\mathbf{f}(Q) = Q \Rightarrow \mathbf{f}^{[n]}(Q) = Q.$$

Proof. Obviously, if $\mathbf{f}(Q) = Q$ then $\mathbf{f}(\mathbf{f}(Q)) = \mathbf{f}(Q) = Q$. We can conclude that then also $\mathbf{f}^{[n]}(Q) = \mathbf{f}^{[n-1]}(\mathbf{f}(Q)) = \mathbf{f}^{[n-1]}(Q) = \cdots = Q$. □

Considering this result, we can conclude that all invariant sets of a dynamical system generated by $\mathbf{f}$ can also be found in a dynamical system generated by $\mathbf{f}^{[n]}$.

Proposition 4.2. *If $Q' \subset M$ is an invariant set for $\mathbf{f}^{[n]}$, $n \in \mathbb{N}$ then $Q = \bigcup_{0 \leq k < n} \mathbf{f}^{[k]}(Q')$ is an invariant set for $\mathbf{f}$, i. e.*

$$\mathbf{f}^{[n]}(Q') = Q' \Rightarrow \bigcup_{0 \leq k < n} \mathbf{f}^{[k]}(Q') = Q = \mathbf{f}(Q).$$

Proof. Note that by definition $\mathbf{f}^{[0]}(\mathbf{x}) = \mathbf{x}$ and, hence, $\mathbf{f}^{[0]}(Q') = Q'$. We split the proof in two parts:

1. $\mathbf{f}(Q) \subseteq Q$: If $\mathbf{x} \in Q \Rightarrow \mathbf{x} = \mathbf{f}^{[k]}(\mathbf{x}')$ for some $\mathbf{x}' \in Q'$ and some $k < n \Rightarrow \mathbf{f}(\mathbf{x}) = \mathbf{f}^{[k+1]}(\mathbf{x}')$. Obviously, $\mathbf{f}^{[k+1]}(\mathbf{x}') \in Q$ because if $k+1 < n$ then $\mathbf{f}^{[k+1]}(\mathbf{x}') \in Q$, and if $k+1 = n$ then $\mathbf{f}^{[k+1]}(\mathbf{x}') = \mathbf{f}^{[n]}(\mathbf{x}') \in Q' \subset Q$.

2. $Q \subseteq \mathbf{f}(Q)$: We first state that $\mathbf{f}^{[k]}(Q') \subseteq \mathbf{f}(Q)$ for each $k < n$ because if $k = 0$ then $\mathbf{f}^{[0]}(Q') = Q' = \mathbf{f}^{[n]}(Q') = \mathbf{f}(\mathbf{f}^{[n-1]}(Q')) \subseteq \mathbf{f}(Q)$, and if $k > 0$ then $\mathbf{f}^{[k]}(Q') = \mathbf{f}(\mathbf{f}^{[k-1]}(Q')) \subseteq \mathbf{f}(Q)$. Next we state that if $\mathbf{x} \in Q \Rightarrow \mathbf{x} \in \mathbf{f}^{[k]}(Q')$ for some k < n. It follows immediately that $\mathbf{x} \in \mathbf{f}^{[k]}(Q') \subseteq \mathbf{f}(Q)$.

□

Proposition 4.3. *If $Q' \subset M$ is an invariant set for $\mathbf{f}^{[n]}$, $n \in \mathbb{N}$ then there is an invariant set Q for $\mathbf{f}$ with $Q' \subseteq Q$, i. e.*

$$\mathbf{f}^{[n]}(Q') = Q' \Rightarrow \exists Q \supseteq Q' : \mathbf{f}(Q) = Q.$$

Proof. The proposition is an immediate conclusion of Proposition 4.2. □

An important conclusion of Proposition 4.3 is that a dynamical system generated by $\mathbf{f}^{[n]}$ still consists of the same invariant sets than the one generated by $\mathbf{f}$. Each invariant set Q' found for $\mathbf{f}^{[n]}$ belongs to an invariant set Q of $\mathbf{f}$. Furthermore, according to Proposition 4.1, all sets Q of $\mathbf{f}$ can be found in the dynamical system of $\mathbf{f}^{[n]}$.

Recall now that most of our investigations aim to detect specific types of invariant sets. If all invariant sets of $\mathbf{f}$ are preserved in the dynamical system of $\mathbf{f}^{[n]}$ then, obviously, they can also be detected in $G_{f^{[n]}}$. However, note that the characteristics of the sets might change. Let us look, for instance, on the invariant sets of periodic points $P(p)$, see Eq. 2.4. The invariant set $P(6)$ of $\mathbf{f}$ is then equivalent to the invariant set $P(2)$ of $\mathbf{f}^{[3]}$ but the points belonging to these sets have a different periodicity with respect to $\mathbf{f}$ and $\mathbf{f}^{[3]}$. Hence, one has to be careful when analyzing the results of $G_{f^{[n]}}$. However, although the periodicity might change, every periodic point of $\mathbf{f}$ is also periodic for $\mathbf{f}^{[n]}$, and no other periodic points than for $\mathbf{f}$ are found for $\mathbf{f}^{[n]}$. The same is true for points belonging to quasiperiodic trajectories (without proof).

Each edge in the graph $G_{f^{[n]}}$ represents a longer part of a trajectory than in G_f. In terms of tuning this is of interest because transient dynamics can then be better distinguished from asymptotic ones. Less cells which do not contain a solution are selected for subdivision, and the growth rate of cells during the subdivision process is lower. However, the tuning has also some drawbacks. First of all, the computation time for the construction of $G_{f^{[n]}}$ increases by factor n in comparison to G_f. Furthermore, it is more likely that unstable parts of the solution, e.g. unstable periodic or quasiperiodic points, might not be detected because the forward iterates $\mathbf{y} = \mathbf{f}^{[n]}(\mathbf{x})$ diverges stronger from these objects than $\mathbf{y} = \mathbf{f}(\mathbf{x})$, see also the discussion in Sec. 4.2.3. Last but not least, taking the analytical point of view, one must be aware about the change of characteristics regarding the invariant sets of $G_{f^{[n]}}$ in G_f.

The practical usage of the tuning method is illustrated by the following example. Let us consider the logistic map

$$\begin{aligned} &x(n+1) = f_l(x(n)), \quad f_l : [0;1] \to [0;1], \\ &f_l(x) = \alpha x(1-x). \end{aligned} \tag{4.6}$$

For this system, the well-known period-doubling bifurcation scenario can be observed. It is formed by a sequence of flip bifurcations, as described for instance in [GT77, Fei78, Fei79].

We consider the area in the vicinity of the first flip bifurcation point $\alpha = 3$. At this point, the fixed point

$$x^* = 1 - \frac{1}{\alpha}$$

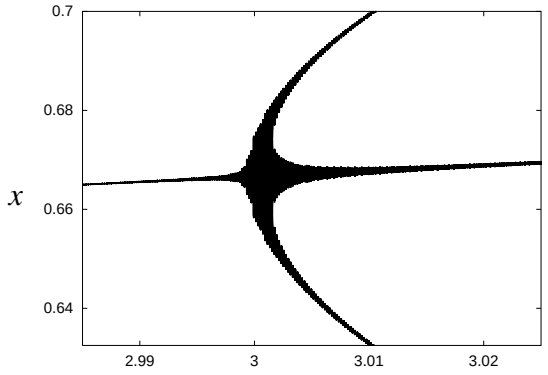
becomes unstable and a two-periodic limit cycle consisting of the points

$$x^{**}_{1,2} = \frac{1}{2} + \frac{1}{2\alpha}\left(1 \pm \sqrt{a^2 - 2a - 3}\right)$$

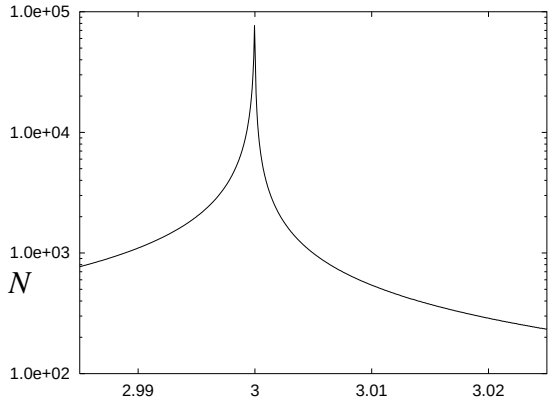
emerges. We assume, that these points should be found using our methods of symbolic analysis. For this reason the interval $M = [0.58; 0.74]$ is divided initially into 50 boxes. In further subdivision steps, the boxes are divided into 4 new ones. In each box 5 scan points are selected.

The results of the calculation for the parameter range $\alpha = [2.98; 3.03]$ are presented in Fig. 4.3(a). As one can see, the points x^* and $x^{**}_{1,2}$ are found, but in the vicinity of the first flip bifurcation point $\alpha = 3$, a large number of boxes are detected as recurrent although it is obvious from the analytical point of view that these boxes correspond to the transient dynamics. This behavior is known as critical slowing down behavior [Gol92] and leads to the blurring of the bifurcation diagram close to the point $\alpha = 3$.

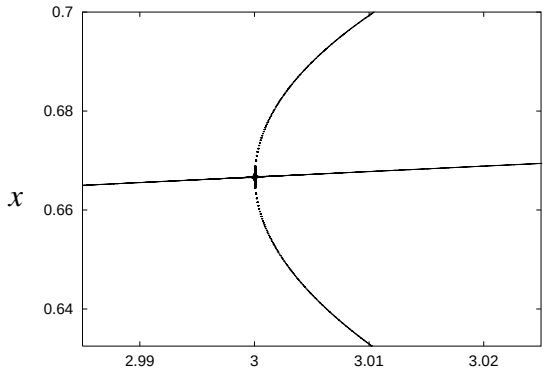
In order to avoid the numerical errors mentioned above, we have to consider the influence of the critical slowing down behavior on the calculation of symbolic images. It is well-known that in the vicinity of a bifurcation point the number of iterations to reach the asymptotic dynamics with a given accuracy grows drastically. In Fig. 4.3(b) this number is shown in a logarithmic representation for the accuracy $\varepsilon < 10^{-8}$. Taking into account the dynamical behavior described above, the blurring of the bifurcation diagram presented in Fig. 4.3(a) can be explained easily. The orbits started in boxes which lie close to fixed points (stable or unstable) do not leave these boxes within a single iteration step. Therefore, the corresponding cells of the symbolic image are marked as one-periodic. In an analogous way, the orbits started in the boxes close to the two-periodic limit



(a) α



(b) α



(c) α

Figure 4.3: *Reduction of the critical slowing down phenomenon by calculation of the bifurcation diagram for the logistic map in the vicinity of the bifurcation point* $\alpha = 3$. *(a) Analytical and numerical results using the function* f_l. *(b) Number of iterations which the system needs in order to reach the asymptotic dynamics, (c) Numerical results using the* 51*-th iterated function* $f_l^{[51]}$.

cycle return into these boxes after two iteration steps, and the corresponding cells are detected as two-periodic. We already described such behavior earlier in this work as clustering. In principle, the problem that an orbit does not leave the box where it was started can take place at any parameter value. However, due to the critical slowing down phenomenon, is becomes more crucial in the vicinity of the bifurcation point.

As one can see, there are two possibilities to improve the quality of the calculation. The first one is to take a covering consisting of smaller boxes than the ones used here. However, in this case the number of boxes grows and the obtained quality improvement is not essential. As a second possibility, we can apply the described tuning method and construct $G_{f^{[n]}}$ for $\mathbf{f} = f_l$. Therefore, let us first consider the fact that a fixed point of the function f_l is obviously a fixed point of the iterated function $f_l^{[n]}$ for any n as well. Secondly, for any even number n a two-periodic limit cycle of the function f_l corresponds to a fixed point of the iterated function $f_l^{[n]}$. For odd numbers n, a two-periodic limit cycle of the function f_l represents a limit cycle with the same period of the iterated function $f_l^{[n]}$. Therefore, the bifurcation diagrams of the function $\mathbf{f}$ and of the higher iterated $f_l^{[n]}$ for any odd n are identical but the trajectories, which do not leave a box within a single iteration step, may do it after a sufficiently large number of steps is reached. Therefore, we can use the iterated function $f_l^{[n]}$ with a sufficiently large odd n instead of the function f_l in order the avoid the critical slowing down behavior which leads to the blurring of the bifurcation diagram. The results for this case are shown in Fig. 4.3.(c), where the 51-st iterated function of the logistic map is used. As one can see, the critical slowing down behavior could not be avoided at all but becomes almost not observable.

4.2.2 Discretization Time for Systems Continuous in Time

For dynamical systems continuous in time, the iteration of the function must be considered in a different context than the iteration of discrete systems. The number of integration steps n, which determines for a fixed integration step size Δt the discretization time $t = \Delta t \cdot n$, is an essential part of the parameter setting. By variation of t one changes the discretization of the continuous trajectory starting from a point $\mathbf{x} \in M(I)$. Taking the theoretical point of view, the ideal approach is to vary the time t for each scan point $\mathbf{x} \in M(I)$ in such a way that the image $\mathbf{f}^{[n]}(\mathbf{x}) = \phi^{[n]}(\mathbf{x}, \Delta t) = \mathbf{y}$ lies in the next neighboring box $M(I')$ to which the continuous trajectory started at $\mathbf{x}$ moves to. By doing so, the complete dynamics of the underlying system can be preserved by the symbolic image graph. Of course, in that context one must also consider and properly treat the case that a

trajectory $\mathbf{x}(t)$ might never leave its initial box. This happens if the box contains an invariant set, whereby it can be assumed for the box size shrinking to zero, that this invariant set is a fixed point.

Unfortunately, empirical experience showed that this approach generally fails in practical application. Even for dynamical systems which exhibit comparatively simple dynamics, like the Van der Pol system, see Sec. 4.1.1, a high level of clustering can be observed and the symbolic image grows too strong in each subdivision step. A way to avoid this is to fix a reasonably large discretization time t for all computations. Due to the fact that longer forward iterates are computed, the same effect happens as for the iteration of the system function described in the last section. Transient dynamics can be better distinguished from asymptotic ones, and less cells are selected for subdivision whose corresponding boxes do not contain a solution. This is in most cases a necessity for a reasonable numerical simulation. Note that the detection of stable parts of a solution is not affected by a high setting of t. However, unstable parts may not be detected because the trajectories started close to them diverge. So it is after all still essential that the symbolic image is constructed by the combination of many short forward iterates instead of a few long ones. Otherwise, the distinctive features of this investigation method can not be used. This means that t must be set to a value so that the symbolic image does not grow too strong but that also the information about the whole solution persists. It is not guaranteed that such a setting exists for every system in focus, and if it exists, there is not yet a general rule how to derive it. Only user experience and heuristic testing can lead to the most proper setting of t.

There are two ways to manipulate t. One is to increase or decrease the number n of iteration steps. Hereby, the precision of the computed parts of the trajectories does not change. Therefore it is the preferred way to control the size of t. However, the performance time for the construction of the symbolic image depends on n, and a high setting can significantly slow down the computation time. Therefore, one can also consider to change the size of the integration step Δt instead of n. This can improve the performance of the calculation. Note that the precision of the computed parts of the trajectories depends on Δt. But as a consequence of the fact that the symbolic image graph is build from small forward iterates, the numerical error which arises from an increase of Δt is by far not as crucial as if long forward iterates would be computed.

For the dynamical systems we tested, settings in the range $t \in [0.1, 0.2]$ turned out to be a good choice. The trajectories are then long enough to detect and exclude many cells which do not contain a solution in an early stage of subdivision.

Nevertheless, the symbolic image can be kept small in the subdivision process. On the other hand, unstable objects are still recognized, as can be seen for the unstable limit cycles of the Lorenz system in Sec. 4.3.1.

4.2.3 Reconstruction of Fragmented Solutions

In the former sections we have discussed the usage of higher iterated functions and large discretization times. In many calculations, these options turned out to be an adequate technique to tune investigation methods. However, it was also mentioned that unstable parts of a solution might not be detected if the number of iterations n or the time t is chosen too large. In practice, we observed, that for crucial settings of the parameters, some unstable invariant sets do not completely disappear at once, but rather more fall apart. Some parts of them are still recognized while others vanish, as it can be seen, for instance, in Fig. 4.5.(a) – (e) for the Lorenz system.

Such a phenomenon is a result of taking only a limited number of scan points per box in combination with following a relatively long run of trajectories in order to construct the edges of the symbolic image graph. This leads to a loss of information about the structure of unstable invariant sets. It is not our intention to give here a detailed analysis of this problem, but rather more a solution for the reconstruction of such unstable objects. Nevertheless, one should keep in mind that not every structure that looks like a disappearing unstable invariant set is necessarily a fragment of the solution. In some cases, it turned out that objects which seemed to be parts of unstable limit cycles belonged to non-cyclic orbits. So after the application of the method of reconstruction, further tests have to be applied to approve the correctness of obtained results.

The method, as introduced here, aims only on the reconstruction of the chain recurrent set. For other investigations, slight changes might be necessary. The reconstruction can be done by application of an extension to the symbolic image construction algorithm. The basic idea here is to add and/or select all cells belonging to boxes $M(I)$ of the symbolic image $G_{f^{[n]}}$ which will be passed by the forward iterates $\mathbf{f}^{[1]}(\mathbf{x}),\ldots,\mathbf{f}^{[n-1]}(\mathbf{x})$ on its way from $\mathbf{x}$ to its image $\mathbf{y}=\mathbf{f}^{[n]}$. Therefore, first the symbolic image $G_{f^{[n]}}$ will be constructed according to the standard approach. Then the investigation method is applied in order to get the set of recurrent cells $RV(G_{f^{[n]}})$. Afterwards, the following extension must be applied before the next subdivision. For every recurrent cell $c_I \in RV(G_{f^{[n]}})$ its corresponding box $M(I)$ is detected. Then, for every scan point $\mathbf{x} \in S(I)$ it has to

be checked, whether its target point $\mathbf{f}^{[n]}(\mathbf{x}) = \mathbf{y} \in \tilde{T}(I)$ lies in a selected cell which is equivalent, i.e. belongs to the same set of strongly connected components. If so, we locate for each value $\mathbf{f}^{[k]}(\mathbf{x})$, $k = 1, \ldots, (n-1)$, the box $\mathbf{f}^{[k]} \in M(I')$. If the box $M(I')$ and its corresponding cell $c_{I'}$ do not exist for a visited area, they will be added to the symbolic image. Furthermore, the cell $c_{I'}$ will be marked as recurrent, no matter if it already existed or was just added.

If this extension is applied, the course of a trajectory, which connects recurrent cells, will be reconstructed. Note that the symbolic image can only become more precise by this extension. If a source cell c_i and its target cell $c_{i'}$ are recurrent and equivalent, then, consequently, all the cells corresponding to boxes which are passed by the connecting trajectory are also recurrent. In fact, if all numerically computed symbolic images of an investigation would have been exact and no approximation, reconstruction would not change them. As already mentioned, this operation might add new cells to the symbolic image. So it can still be applied in a stage of subdivision when the fragmented invariant set has already fallen apart to a large extent. This can be seen in Fig. 4.5, where unstable limit cycles of the Lorenz system will be reconstructed in the 10-th subdivision step. In the following section, some numerical case studies are presented which illustrate the application of the reconstruction method.

4.3 Numerical Case Studies

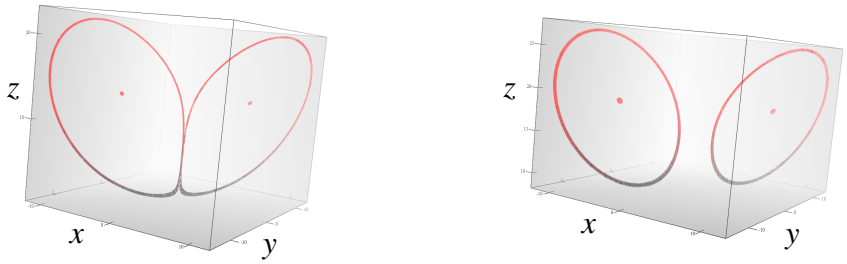
We present two numerical case studies in order to demonstrate the application of the above mentioned tunings. As examples, we chose a system continuous in time and one which is discrete in time. Both of them are 3-dimensional and, hence, they require higher computational resources than the 2-dimensional examples we investigated so far. For this reason, it is necessary to apply tuning methods.

4.3.1 Lorenz System

We consider the well-known dynamical system continuous in time introduced by Lorenz in [Lor63] and defined by

$$\begin{aligned} &\dot{\mathbf{x}}(t) = \mathbf{F}_L(\mathbf{x}(t)), \quad \mathbf{F}_L : \mathbb{R}^3 \to \mathbb{R}^3, \quad \mathbf{x} = (x,y,z)^T \\ &\mathbf{F}_L(\mathbf{x}) = \begin{pmatrix} \sigma(y-x), \\ x(r-z)-y, \\ xy-bz. \end{pmatrix} \end{aligned} \tag{4.7}$$

We use the standard values of the parameters $\sigma = 10$, $b = 8/3$ and investigate the Lorenz system at two values of the parameter r, namely $r_1 \approx 14.6$ and



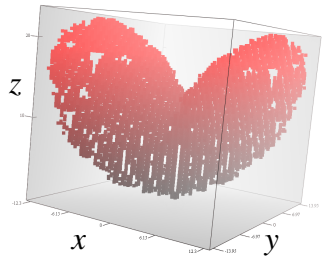
(a) $r_1 = 14.6$ (b) $r_2 = 20$

Figure 4.4: *Lorenz system: Computation of an outer covering of the chain recurrent set at positions* $r_1 = 14.6$ *and* $r_2 = 20$.

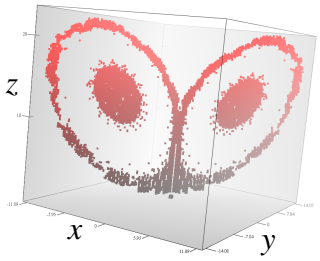
$r_2 \approx 20$. As shown in [Spa73], for these settings exist an unstable fixed point $P = (0,0,0)^T$ and two stable ones C_1 and C_2, each of them accompanied by an unstable limit cycle. The value r_1 is chosen close to the so-called homoclinic explosion which occurs at $r \approx 13.926$, where the unstable manifolds of P return to the origin. Furthermore, at parameter value r_2, the both unstable limit cycles around C_1 and C_2 are situated close to each other and to C_1 and C_2.

In order to reproduce these results with methods of symbolic analysis, we compute the chain recurrent set. We define for r_1 and r_2 the domain spaces $M_1 = [-35.0;35.0] \times [-35.0;35.0] \times [0.0;30.0]$ and $M_2 = [-20.0;20.0] \times [-20.0;20.0] \times [0.0;30.0]$ as the area of investigation. The division of these spaces is initially set to $4 \times 4 \times 2$ and $2 \times 2 \times 2$ boxes. In the following subdivision stages each box is divided into $2 \times 2 \times 2$ smaller boxes. The integration step Δt is set to 0.001, and the number of iteration steps to $n_1 = 100$, $n_2 = 200$. In order to compute the integration step $\phi(\Delta t, \mathbf{x})$, the Runge-Kutta method was applied.

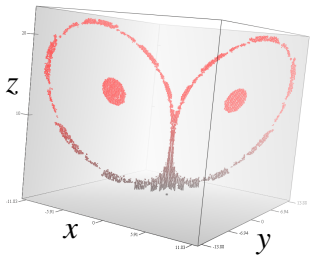
Figs. 4.4(a) and 4.4(b) show the results of the calculations for the parameters r_1 and r_2. Remarkably, one can see that the limit cycles for r_1 still touch each other, which is due to some numerical inaccuracy, while for r_2 the cycles shrinked closer around C_1 and C_2. The computations took 30 minutes for r_1 and 2 hours for r_2. Ten subdivision steps were computed, and the symbolic images contained up to 1 400 000 cells. Hereby, the high computation time is mainly due to the relative high setting of the iteration time t. Furthermore, the unstable fixed point



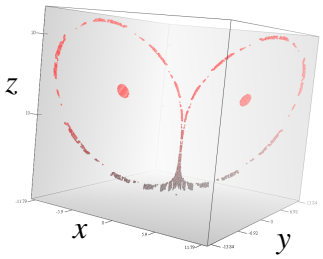
(a) fifth subdivision step



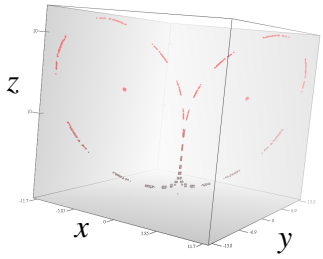
(b) sixth subdivision step



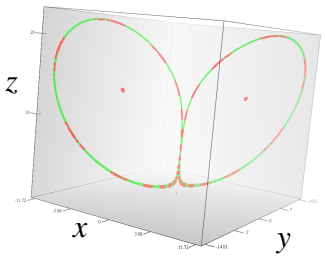
(c) seventh subdivision step



(d) eighth subdivision step



(e) tenth subdivision step



(f) completion of the fragmented limit cycle

Figure 4.5: *Lorenz system: Reconstruction of unstable limit cycles at parameter* $r_1 = 14.6$ *with a large discretization time. The limit cycles fall apart and vanish by time (red), but will be completed (green).*

P can not be computed by this setting. However, if t would be set to a lower value, the limit cycles could not be detected at all because too many cells would be selected for subdivision and the memory resources would be exceeded after a few subdivisions.

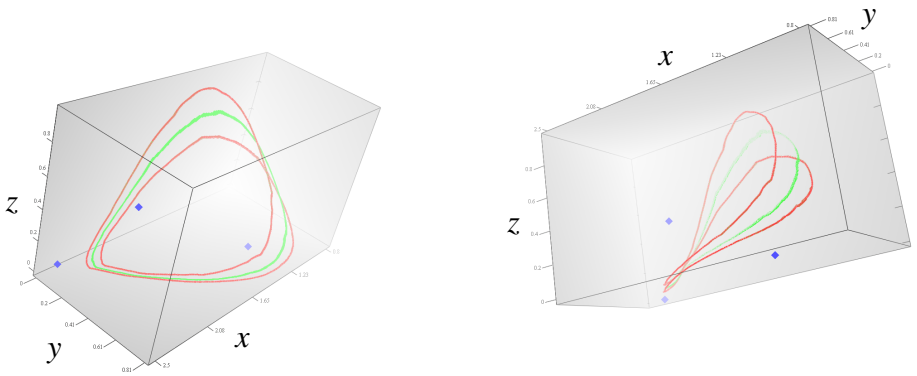
Several subdivision steps for the parameter setting $r_1 = 14.6$ are illustrated by the Figs. 4.5(a–f). We see that the principal shape of the cycles becomes visible in the fifth subdivision step, while the distinction into fixed points and cycles is visible in the sixth subdivision. Note that if t would be set to a smaller value, this distinction could not be computed by our methods. Too many cells would be considered recurrent and the size of the symbolic image would be too big for further calculations after the sixth or seventh subdivision step. In Figs. 4.5(c–e) we see the computations for the next subdivision steps. Although the high setting of t allows the computation of the distinct invariant sets, a side effect is that parts of the unstable limit cycles get lost. For this reason, the method for reconstruction of the fragmented solutions must be applied. The results are shown in Fig. 4.5(f). We see that the final computation produces a precise outer covering of the unstable limit cycles.

4.3.2 Discrete Food Chain Model

Next we analyzed a discrete system of mathematical biology. The 3-dimensional dynamical model describes a discrete food chain model, studied by Lindström in [Lin02]. The system is defined by

$$
\begin{aligned}
&\mathbf{x}(n+1) = \mathbf{f}_{dfc}(\mathbf{x}(n)),\\
&\mathbf{f}_{dfc} : \mathbb{R}^3 \to \mathbb{R}^3, \quad \mathbf{x} = (x,y,z)^T \\
&\mathbf{f}_{dfc}(\mathbf{x}) = \begin{pmatrix} \dfrac{\mu_0\, x\, e^{-y}}{1 + x \max\left(e^{-y}, g(z)g(y)\right)} \\ \mu_1\, x\, y\, e^{-z} g(y) g(\mu_3\, y\, z) \\ \mu_2\, y\, z \end{pmatrix}, \\
&\text{with} \quad g(s) = \begin{cases} \dfrac{1-e^{-s}}{s}, & \text{if} \quad s \neq 0, \\ 1, & \text{if} \quad s = 0. \end{cases}
\end{aligned}
\tag{4.8}
$$

In this section, we will only focus on the following parameter setting: $\mu_0 = 3.4001$, $\mu_1 = 1$ and $\mu_2 = \mu_3 = 4$.



(a) (b)

Figure 4.6: *Discrete food chain model: Two different views of the outer covering of the chain recurrent set. The attractor (red), an unstable quasiperiodic cycle (green) and the unstable fixed points (blue) are shown. The fourth fixed point at* $(0,0)$ *can not be seen.*

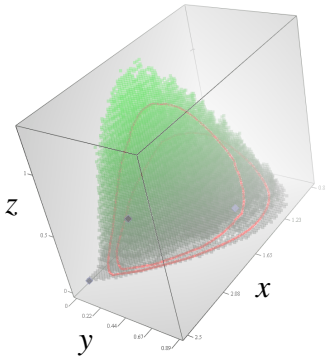
The analytic results of Lindström showed, that Eq. 4.8 possesses at most four fixed points and that three of them can be given analytically by:

$$\begin{aligned} P_0 &= (0,0,0)^T \\ P_1 &= (\mu_0 - 1, 0, 0)^T \\ P_2 &= \left(\frac{\mu_0 \log\left(\frac{\mu_1\mu_0}{1+\mu_1}\right)}{(\mu_0-1)\mu_1 - 1}, \log\left(\frac{\mu_1\mu_0}{1+\mu_1}\right), 0 \right)^T \end{aligned} \tag{4.9}$$

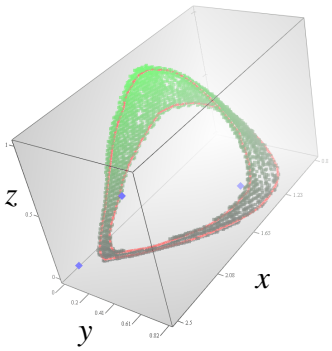
By our investigation methods, the following fixed points can be approximated:

$$\begin{aligned} \tilde{P}_0 &= (0,0,0)^T, \\ \tilde{P}_1 &= (2.4001,0,0)^T, \\ \tilde{P}_2 &= (1.289,0.531,0)^T, \\ \tilde{P}_3 &= (2.116,0.25,0.423)^T. \end{aligned} \tag{4.10}$$

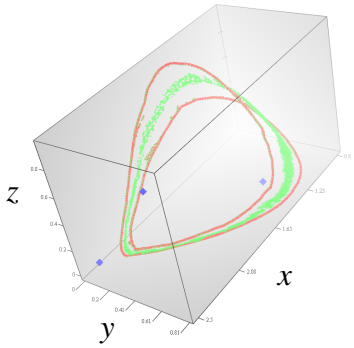
However, our main intention is not the localization of some fixed points, but rather more the computation of the complete chain recurrent set within the area $M = [-1.0; 4.0] \times [-1.0; 4.0] \times [0.0; 1.6]$. In this case, it was not possible to get an appropriate approximation by means of usual symbolic image construction.



(a) second subdivision step



(b) third subdivision step



(c) fourth subdivision step

Figure 4.7: *Discrete food chain model: Numerically calculated fixed points and three subdivision steps of the symbolic image construction. The outer covering of the chain recurrent set (green) as well as the attractor (red) are shown. Note that in this example the attractor was computed by forward iterates.*

The tuning techniques must be applied to get satisfiable results. By doing so, the equilibrium points, and maybe some other information, get lost in the symbolic image after several subdivision steps. On the other hand, two invariant manifolds can be detected which belong to different components of the chain recurrent set, see Fig. 4.6. By application of forward iteration, it can be verified that both of them consist of quasiperiodic trajectories, and that one is a stable invariant set, namely an attractor (red), while the other is an unstable invariant set (green). Hereby, the unstable entity is not a repeller but of saddle type. For this reason, it could not be approximated by backward iterates. Such a calculation takes around one hour and the symbolic image grows up to $\approx 1\,100\,000$ cells. The long calculation time is mainly caused by the application of the tuning-techniques. Note that the localization of the unstable quasiperiodic manifold is, from the computational point of view, a nontrivial task. To the authors' knowledge, no other numerical computation method is able to detect this entity. This is, among others, due to the fact, that the system described by Eq. 4.8 is only piecewise-smooth, has no explicit inverse, and possesses dynamics which are, in general, difficult to handle, also for other set-oriented and symbolic image-like approaches. Only by application of the tuning methods the results can be computed.

In order to get a better impression how the construction process works, Fig. 4.7 shows the results of several subdivision steps. Hereby, 17 scan points per box are taken. The rough position of the attractor can be located after the second subdivision of the domain space M into $200 \times 200 \times 32$ regions, see Fig. 4.7(a), then, in the third subdivision, see Fig. 4.7(b), the principal shape of the attractor becomes visible. But only after the fourth subdivision into $1\,200 \times 1\,200 \times 192$ regions, see Fig. 4.7(c), the symbolic image splits into two different sets of equivalent recurrent cells, which correspond to the stable and unstable invariant manifolds. In order to achieve these results, it is necessary to compute the symbolic image graph for the iterated function $\mathbf{f}^{[40]}$ in the third subdivision and for $\mathbf{f}^{[80]}$ in the fourth subdivision step. Otherwise, the principal shape of the cone, see Fig. 4.7(a), would persist during further subdivisions. Additionally, reconstruction of the fragmented parts must be applied in order to avoid that the cycles vanish. The final result, see Fig. 4.6, is computed after the sixth subdivision. Note that in the subdivisions 5 and 6, also the function $\mathbf{f}^{[40]}$ is used and reconstruction of the cycles applied.

Chapter 5

Investigation of the Symbolic Image

The main subject of the last two chapters was an efficient implementation of the basic techniques of symbolic analysis. We discussed the construction of the symbolic image graph as well as several extensions and tunings which can be applied on it. In that context, only basic investigations of the symbolic image graph were introduced, namely the localization of the chain recurrent set and of periodic points. These two techniques already proved to be powerful tools for the global analysis of dynamical systems. In this chapter, we propose now several more approaches which allow a more sophisticated investigation of the symbolic image graph. These approaches are based on concepts of symbolic analysis. Although the theoretical basics of these concepts are already settled, several more steps of development are necessary for a practical application. It is our aim to close this gap and provide methods which are applicable in practice for all fields of investigation mentioned in Chapter 2.

The results of the last chapters build the foundation for the theoretical considerations and algorithms proposed here. An efficient construction and subdivision of the symbolic image graph are required for all techniques. The theoretical background for the investigations was set by Osipenko [Osi04]. In the context of a practical application, some aspects of this theory must be extended. Furthermore, we provide the required algorithms and prove their correctness. Several numerical examples illustrate the application of the concepts in practice.

5.1 Localization of Attractors and Their Basins

At first we present an algorithm for the localization of an attractor and its basin. The task of this algorithm is to detect the cells in a symbolic image G which correspond to the boxes covering an attractor and its basin in the state space. In

order to do so, we start by giving a definition of attractors and basins on a symbolic image, and point out their correspondence with real attractors and basins.

5.1.1 Attractors on a Symbolic Image

Consider a symbolic image G of the continuous mapping $\mathbf{f}$. A set of cells $L \subset V(G)$ generates a subgraph $G(L)$ which contains the cells L and the edges $c_i \to c_j$ iff the cells c_i and c_j belong to L.

Definition 5.1. [Osi99] *A set $L \subset V(G)$ is invariant if for each cell $c_i \in L$ an admissible path through c_i is in L.*

So we say that the set L is invariant if for each cell $c_i \in L$ the edges $c_j \to c_i$ and $c_i \to c_k$ exist in $G(L)$.
The set of cells

$$En(L) = \{c_j \in L : \text{ there exists an edge } c_i \to c_j,\ c_i \notin L\} \tag{5.1}$$

is called the entrance of L and

$$Ex(L) = \{c_i \in L : \text{ there exists an edge } c_i \to c_j,\ c_j \notin L\} \tag{5.2}$$

is called the exit of L.

Definition 5.2. [Osi99] *An invariant set L is an attractor of a symbolic image if $Ex(L) = \emptyset$. It is a repeller if $En(L) = \emptyset$.*

The following proposition describes the structure of an attractor on G.

Proposition 5.1. [Osi99] *Each attractor L consists only of some classes of equivalent recurrent cells and all paths between these classes.*

Let L be an attractor. A basin or domain of attraction is the set of vertices

$$D(L) = \{c_i \mid \text{each path through } c_i \text{ finishes in } L\}, \tag{5.3}$$

i.e., for each path $\{\ldots, c_i, \ldots, c_{j_k}, \ldots\}$ there exists a k^* so that the vertices c_{j_k} with $k > k^*$ belong to L.

Proposition 5.2. [Osi99] *Let L be an attractor of a connected symbolic image G. Then*

1. *the cells from $D(L) \setminus L$ are non-recurrent,*
2. *the set of cells $L^* = V(G) \setminus D(L)$ is a repeller.*

Obviously, this proposition requires that the graph G is connected.

It was shown in Osipenko [Osi99] that, in theory, an attractor of a dynamical system and its domain of attraction can be constructed as precisely as one likes by a symbolic image. Therefore, let us consider a covering C of the phase space and the corresponding symbolic image G. We pick an attractor L on G and detect the sets

$$A(L) = \left\{ \bigcup M(i), c_i \in L \right\}, \tag{5.4}$$

$$W(L) = \left\{ \bigcup M(j), c_j \in D(L) \right\}, \tag{5.5}$$

$$R(L) = \left\{ \bigcup M(k), c_k \in L^* \right\}. \tag{5.6}$$

Theorem 5.1. [Osi99] *If L and $D(L)$ are an attractor and its basin on a symbolic image G, then there is an attractor Λ of $\mathbf{f}$ and its basin $W^s(\Lambda)$ such that*

1. *the set $U = int \cdot A(L)$, where $int\cdot$ denotes the interior, is a neighborhood of Λ such that $\mathbf{f}(cl\, U) \subseteq U$,*
2. *the set $W(L)$ is an estimation of the basin $W^s(\Lambda)$ so that $W(L) \subset W^s(\Lambda)$.*

Obviously, by detecting an attractor and its basin on a symbolic image, a real attractor Λ and its basin $W^s(\Lambda)$ can be approximated. In order to improve the approximation, a multilevel subdivision scheme can be applied. The following outline of an algorithm was proposed in Osipenko [Osi99]:

1. Set $s \Leftarrow 0$, and construct an initial covering C^0.
2. The symbolic image G^s is built for a covering C^s.
3. An attractor L^s on G^s is selected by the user.
4. The sets $A(L^s), W(L^s), R(L^s)$ are detected.
5. The boxes $M(i)$ belonging to $A(L^s)$ and $R(L^s)$ are subdivided. Note that it is required that the boxes $W(L^s)$ are neither deleted nor subdivided.
6. A new covering C^{s+1} is constructed for the new phase space discretization.
7. Set $s \Leftarrow s+1$ and return to the second step.

Let $d(C) = \max(\delta(M(i)) | c_i \in (L \cup L^*))$ be the maximal diameter of the cells belonging to the attractor and repeller on G. We can state the following theorem.

Theorem 5.2. [Osi99]

1. *The described algorithm gives a sequence of embedded sets*

$$\begin{aligned} A(L^1) &\supseteq A(L^2) \supseteq \ldots, \\ W(L^1) &\subseteq W(L^2) \subseteq \ldots, \\ R(L^1) &\supseteq R(L^2) \supseteq \ldots. \end{aligned} \tag{5.7}$$

2. *If* $d(C^s) \to 0$ *as* $s \to \infty$ *then*
   $\lim_{s\to\infty} A(L^s) = \Lambda$ *is an attractor,*
   $\lim_{s\to\infty} W(L^s) = W^s(\Lambda)$ *is its domain of attraction,*
   $\lim_{s\to\infty} R(L^s) = \Lambda^*$ *is the dual repeller.*

3. *Any attractor* Λ *can be constructed by such an algorithm.*

Our implementation will mainly follow these propositions of Osipenko though some modifications are necessary for a practical application as will be pointed out later.

Remark 5.1. *In the original work [Osi99] a homeomorphism is assumed as the underlying dynamical system for Theorem 5.2 and the proposed approximation of attractors and basins by multilevel subdivision. However, the concepts are also valid for noninvertible mappings. Reason for this is that the inverse of the system function is not required for any of the proofs. Hence, we assume that the proposed concepts can be applied on all dynamical systems generated by continuous mappings, no matter if an inverse exists or not. The only restriction to be aware of is, that the definition of a repeller, see Def. 2.7, is limited to homeomorphisms. Hence, in this context the repeller of a symbolic image only approximates a real repeller, if* $\mathbf{f}$ *is a homeomorphism.*

5.1.2 Construction of the Acyclic Graph *DG*

The symbolic image G is represented by a cyclic directed graph. We transform G into a different graph representation which is more suitable for our needs. Thus we construct from G a *directed acyclic graph* (dag) DG. Recall that each H_k represents a set of equivalent recurrent cells, see Def. 3.3. In DG all cells $c_i \in H_k$ are merged to a new, single cell $\tilde{c}_k$ which has all the in- and outgoing edges of the cells belonging to the recurrent cell set H_k. Hence, the graph DG consists only of non-recurrent cells which either represent all equivalent cells of a set H_k or one of the non-recurrent cells of G. Obviously, there are also no self-connecting edges $\tilde{c}_k \to \tilde{c}_k$ in DG. Furthermore, we will assign to every *forward edge* $\tilde{c}_i \to \tilde{c}_j$ a *backward edge* $\tilde{c}_i \leftarrow \tilde{c}_j$ so that we can follow paths on DG in both directions –

forwards and backwards.

In the following, we denote a cell belonging to DG as a dag cell $\tilde{c}_i$. We can consider that the graph DG has two subsets. Firstly, the dag cells representing a set of equivalent recurrent cells,

$$DV(\zeta) = \{\tilde{c}_k \,|\tilde{c}_k \text{ is a dag cell for } H_k\}, \tag{5.8}$$

and, secondly, the dag cells which correspond to non-recurrent cells in G,

$$DV(G) = \{\tilde{c}_l \,|\tilde{c}_l \text{ is a dag cell for } c_l \in V(G) \setminus RV(G)\}. \tag{5.9}$$

Note that $DV(\zeta)$ and $DV(G)$ are disjunct. The union of these sets are the vertices of the new graph DG:

$$V(DG) = DV(\zeta) \cup DV(G). \tag{5.10}$$

We also assume references between the cells of G and DG according to the following definition.

Definition 5.3. *A cell $\tilde{c}_k \in V(DG)$ is called corresponding to $c_i \in V(G)$ if it is the dag cell which represents c_i in DG.*

The corresponding dag cell $\tilde{c}_k$ for a cell c_i,

$$d(c_i) = \tilde{c}_k,\ d : G \mapsto DG,$$

is either a cell $\tilde{c}_k \in DV(G)$ if c_i is non-recurrent or, if c_i is recurrent, the cell $\tilde{c}_k \in DV(\zeta)$ for the set H_k to which c_i belongs to. The mapping d is considered to be surjective.

The forward edges between the cells of DG are represented as adjacency lists. More precisely, for each $\tilde{c}_i \in V(DG)$ there is a list,

$$E(\tilde{c}_i) = \{\tilde{c}_j \mid \text{there exists an edge } \tilde{c}_i \to \tilde{c}_j\}. \tag{5.11}$$

Additionally, there are adjacency lists for the backward edges. To each $\tilde{c}_i$ belongs a list of parent cells

$$P(\tilde{c}_j) = \{\tilde{c}_i \mid \text{there exists an edge } \tilde{c}_i \to \tilde{c}_j\}. \tag{5.12}$$

Note that the distinction between the graphs G and DG should only be considered in the theoretical context. For an implementation, it is not necessary to explicitly construct DG. Rather more, the construction of G can be extended so that G implicitly contains the information of DG.

5.1.3 Selection of an Attractor L

For the localization of basins, it is necessary to pick an attractor L. In our implementation, the user of the software is able to select for every symbolic image G subsets $S_H \subseteq \zeta$, so that the union of the sets in S_H form an invariant set L_S,

$$L_S = \bigcup_{H_k \in S_H} H_k. \tag{5.13}$$

Such selections S_H can either be defined by a list of indices $\sigma = \{k\}$, or by a region $R \subset M$ in the domain space M so that the set

$$S_H(\sigma) = \{H_k \,|k \in \sigma\} \text{ or} \tag{5.14}$$
$$S_H(R) = \{H_k \,|r(H_k) \cap R \neq \emptyset\} \tag{5.15}$$

can get selected. In this context, $r(H_k)$ is defined as the range of the boxes $M(i)$ belonging to the cells $c_i \in H_k$ in the domain space M.

The selection L_S is an invariant set but not yet an attractor because there is no guarantee that $Ex(L_S) = \emptyset$, see Def. 5.2. However, our implementation allows the localization of the basin $D(L_S)$. Reason for this is to give the user more freedom of selection. Later we will show ways how to check if $Ex(L_S) = \emptyset$ and to construct an attractor L for such selections, see Sec. 5.3. In spite of that it is also possible to set an additional flag so that only sets H_k with $Ex(H_k) = \emptyset$ are selected in the region R and the user can acquire the sets

$$\tilde{S}_H = \{H_k | r(H_k) \cap R \neq \emptyset \text{ and } Ex(H_k) = \emptyset\}, \tag{5.16}$$
$$\tilde{L} = \bigcup_{\tilde{S}_H} H_k. \tag{5.17}$$

It follows immediately that $Ex(\tilde{L}) = \emptyset$ for $\tilde{S}_H$.

An implementation of these concepts is quite easy. If S_H is defined by a list σ then we select every set $H_k \in \zeta$ with index $k \in \sigma$. In case of a region selection R the algorithm tests for every recurrent cell $c_i \in RV(G)$ if $M(i) \cap R \neq \emptyset$ and, if so, the set H_k with $c_i \in H_k$ gets selected. To get the selection $\tilde{S}_H$ an additional check on every H_k in S_H is necessary. The cell $\tilde{c}_k \in DV(\zeta)$ on DG corresponding to H_k is located. Then the algorithm tests if the list of its target cells $E(\tilde{c}_k)$ is empty, i.e. $|E(\tilde{c}_k)| = 0$. If so, $\tilde{c}_k$, and so also the set H_k, have no exit edge. Only such sets H_k get selected for $\tilde{S}_H$.

5.1.4 Localization of the Domain of Attraction

The localization of the basin involves two steps. At first the *upper bound* of the basin will be detected and then the *lower bound* on DG. In this work the upper bound is defined as the set

$$D_u(L) = \{c_i \mid \text{there is a path through } c_i \text{ which finishes in } L\}, \tag{5.18}$$

and the lower bound on DG as the cells belonging to

$$\tilde{D}(L) = \{c_i \mid \text{each path on } DG \text{ through } d(c_i) \text{ finishes in a } d(c_l), c_l \in L\}. \tag{5.19}$$

Furthermore, $D(L)$, see Eq. 5.3, is denoted as the lower bound on G. This implies the following relation:

$$D_u(L) \supseteq \tilde{D}(L) \supseteq D(L).$$

Here, $D_u(L) \supseteq \tilde{D}(L)$ is trivial and $\tilde{D}(L) \supseteq D(L)$ implies that the set $\tilde{D}(L) \setminus L$ can have recurrent cells, compare with Prop. 5.2. Also consider that this definition implies that the repeller L^* of L might intersect with the lower bound, $L^* \cap \tilde{D}(L) \neq \emptyset$.

The upper bound can be located on DG by applying a breadth-first search. As shown in Alg. 5.1, we start the search for every marked cell $\tilde{c}_k \in DV(\zeta)$ whose corresponding cells c_i are in L or, respectively, $H_k \in S_H$. Note that all edges are traversed backwards, or, in other words, for every cell $\tilde{c}_i$ which is popped out of the waiting queue Q, the list of parent cells $P(\tilde{c}_i)$ instead of the target cells $E(\tilde{c}_i)$ will be referred to get the next edges of our search. In the following, we will denote this kind of operation as a *reverse breadth-first search*. All cells $\tilde{c}_i$ which are visited get a mark. All cells $c_i \in G$ corresponding to such marked cells $\tilde{c}_i$ belong to $D_u(L)$.

Theorem 5.3. *Algorithm 5.1 locates all cells belonging to $D_u(L)$ and terminates.*

Proof. A breadth-first search started at a vertex $\tilde{c}_k$ of a directed graph visits every cell $\tilde{c}_j$ of the graph with a path

$$\tilde{c}_k \rightarrow \cdots \rightarrow \tilde{c}_j.$$

Remember now that the search is applied on the backward edges of DG. If there is a backward edge, there is also a forward edge and, hence, we visit all cells $\tilde{c}_p$ in DG which have a path to $\tilde{c}_k$. If there is a path on DG from a cell $\tilde{c}_p$ to $\tilde{c}_k$ then there is also a path on G from each cell c_p with $d(c_p) = \tilde{c}_p$ to the cells c_i with $d(c_i) = \tilde{c}_k$ because all equivalent cells in a set H_k have a path to each other. No other cells c_j on G can have a path to a cell $c_i, d(c_i) = \tilde{c}_k$. We show

Algorithm 5.1 Calculates $D_u(L)$. The algorithm applies a reverse breadth-first search for every $\tilde{c}_k$ corresponding to a $H_k \in S_H$.

Require: DG, S_H with $L = \bigcup_{H_k \in S_H} H_k$
Ensure: All cells $\tilde{c}_i \in V(DG)$ corresponding to cells $c_i \in D_u(L)$ get marked

```
1: for all H_k ∈ S_H do
2:    Locate the cell c̃_k ∈ DV(ζ) corresponding to H_k
3:    Create an empty waiting queue Q
4:    Push c̃_k in Q
5:    while Q is not empty do {Start reverse breadth-first search from c̃_k}
6:       Pop next c̃_i out of Q
7:       Mark c̃_i
8:       for all c̃_j ∈ P(c̃_i) do {Traverse over all parent cells of c̃_i}
9:          if c̃_j was not yet visited then
10:            Push c̃_j in Q
11:         end if
12:      end for
13:   end while
14: end for
```

this by contradiction. Imagine there is a cell c_j which has a path to $c_i, d(c_i) = \tilde{c}_k$ and $\tilde{c}_j = d(c_j)$ is not connected by a path with $\tilde{c}_k$. Then $\tilde{c}_j$ is not visited by the reverse breadth-first search. But this is a contradiction because all edges of G except the ones between equivalent recurrent cells are preserved in DG. Hence, such a cell c_j can not exist and we locate all cells on G which have a path to cells $c_i, d(c_i) = \tilde{c}_k$, if we visit all cells $\tilde{c}_p$ which are connected by backward edges with $\tilde{c}_k$.

The breadth-first search is started for every cell corresponding to a set $H_k \in S_H$ with $L = \bigcup_{H_k \in S_H} H_k$. Therefore we visit every cell of the upper bound. If a cell is visited first time it is pushed in Q. Every cell in Q is eventually popped out and, hence, get marked. So every cell of the upper bound gets marked.

The algorithm terminates if the `while`-loop terminates. This loop depends on the waiting queue Q. Q grows if a cell $\tilde{c}_i$ is pushed in. For each $\tilde{c}_i$ this can happen only once – when it is visited first time. Hence, the algorithm terminates. □

If the upper bound of a basin has been located then the next step is the detection of the lower bound. For simplification, we denote that a cell $\tilde{c}_i$ belongs to $D_u(L)$ (or $\tilde{D}(L)$) if its corresponding cell(s) c_i belong to $D_u(L)$ (or $\tilde{D}(L)$). Detection of the lower bound can be achieved by starting a reverse breadth-first search for every

Algorithm 5.2 Calculates $\tilde{D}(L)$.

Require: DG, S_H with $L = \bigcup_{H_k \in S_H} H_k$
Ensure: All cells $\tilde{c}_i \in V(DG)$ corresponding to cells $c_i \in \tilde{D}(L)$ get marked

```
1: Mark all cells corresponding to D_u(L) { Apply algorithm 5.1}
2: for all c̃_k ∈ V(DG) do
3:   if c̃_k is unmarked then
4:     Create an empty waiting queue Q
5:     Push c̃_k in Q
6:     while Q is not empty do {Start reverse breadth-first search from c̃_k}
7:       Pop next c̃_i out of Q
8:       for all c̃_j ∈ P(c̃_i) do {Traverse over all parent cells of c̃_i}
9:         if c̃_j was not yet visited then
10:          Push c̃_j in Q
11:          if c̃_j is marked then {c̃_j ∈ D_p(L)}
12:            Unmark c̃_j
13:          end if
14:        end if
15:      end for
16:    end while
17:  end if
18: end for
```

cell $\tilde{c}_k$ which got not marked as part of the upper bound, $\tilde{c}_k \notin (D_u(L) \cup L)$, see Alg. 5.2. Every cell $\tilde{c}_p$ visited by traversing over the backward edges of each $\tilde{c}_k$ has at least one path which does not lead to L and, hence, it can not belong to the lower bound,

$$\tilde{c}_p \in V(DG) \setminus \tilde{D}(L).$$

The set of cells of the upper bound which were visited by this operation,

$$D_p(L) = \{\tilde{c}_i | \tilde{c}_i \in (V(DG) \setminus \tilde{D}(L)) \cap D_u(L)\}, \tag{5.20}$$

will be unmarked because they are considered as not being part of the lower bound. The remaining marked cells belong then to the lower bound:

$$\tilde{D}(L) = D_u(L) \setminus D_p(L). \tag{5.21}$$

Note that in our implementation the breadth-first search will not traverse parent cells $\tilde{c}_p \in L$. Such a parent cell can exist if L is an invariant set but not an attractor. For simplification we do not consider this special case in Alg. 5.2.

Theorem 5.4. *Algorithm 5.2 locates the lower bound on DG.*

Proof. Remember that $\tilde{D}(L) \subseteq D_u(L)$. Every cell $\tilde{c}_i$ in $D_u(L)$ which has a path to a cell $V(DG) \setminus D_u(L)$ will be unmarked. All remaining marked cells $\tilde{c}_j$ do not have a path to an unmarked cell. Hence, each path going through such a cell $\tilde{c}_j$ does not leave $D_u(L)$ and, necessarily, finishes in L. So all cells still marked belong to $\tilde{D}(L)$. □

In order to get $D(L)$, the above algorithm needs a slight extension. After the upper bound was marked, all cells $\tilde{c}_u \in DV(\zeta) \setminus L$ will be unmarked and we get the set $D_u^*(L) \subseteq D_u(L)$. If the localisation of the lower bound is applied on the remaining marked cells $D_u^*(L)$ instead of $D_u(L)$ then every cell in $D_u^*(L)$ which has a path to a recurrent cell $c_i \notin L$ will not be considered as being part of the lower bound and we get $D_l^*(L) = D(L)$.

Theorem 5.5. *The algorithm as described above locates the basin $D(L)$.*

Proof. If we unmark all cells of $D_u(L)$ which are recurrent cells not belonging to L then no cells of $D(L)$ get unmarked because all cells in $D(L)$ are non-recurrent, see Prop. 5.2. It follows that $D(L) \subseteq D_u^*(L)$ and also $D(L) \subseteq D_l^*(L)$. If we calculate $D_l^*(L)$, all non-recurrent cells which have a path to other invariant sets than those belonging to L will be unmarked. All remaining marked cells have only paths to cells in $H_k \subseteq L$ and, consequently, are cells of the basin $D(L)$. Hence, $D_l^*(L) \subseteq D(L) \Rightarrow D_l^*(L) = D(L)$. □

5.1.5 Subdivision of the Domain of Attraction

We showed algorithms for the localization of a basin of a symbolic image G. Next it is important to consider how these algorithms can be combined with multilevel subdivision. Let s be the level of subdivision. It was stated before that $W(L^s)$ is the area in the domain space M which is covered by boxes belonging to the basin of an attractor L^s, see Eq. 5.4. Note now that $W(L^s)$ grows in every subdivision s, see Theorem 5.2:

$$W(L^1) \subseteq W(L^2) \subseteq \ldots$$

Such behavior is not desired for our calculations. According to the multilevel subdivision scheme as introduced in Secs. 3.1 and 3.2, we have the following relations for the coverings C^s of a subdivision cascade:

$$C^1 \supseteq C^2 \supseteq \ldots.$$

By investigation of the symbolic images G^s, an outer covering of the solution must be detected which shrinks by subdivision. An area which belongs to a box $M(i)$ of an unselected cell c_i in G^s is discarded, and not part of the investigation in later subdivisions G^{s+k}, $k \in \mathbb{N}$. This contradicts with the growth

of $W(L^s)$. The approximation $W(L^s)$ is not an outer covering of the solution, but a subset of $W^s(\Lambda)$, see also Theorem 5.1. Hence, within our framework of multilevel subdivision, a sequence $W(L^1), W(L^2), \ldots$ can not be directly computed as proposed in the original algorithm of Osipenko [Osi99], see Theorem 5.2.

Let us therefore look at the following set for the upper bound $D_u(L)$,

$$W_u(L) = \left\{ \bigcup M(i), c_i \in D_u(L) \right\}. \tag{5.22}$$

Proposition 5.3. *If L is an attractor on a symbolic image and the set $D_u(L)$ an upper bound of its basin then there are an attractor Λ and its basin $W^s(\Lambda)$ such that the set $W_u(L)$ is an outer covering of $W^s(\Lambda)$, i.e. $W^s(\Lambda) \subseteq W_u(L)$.*

Proof. The existence of Λ for L was already stated in Theorem 5.1. So we only need to prove that $W_u(L)$ is an outer covering of the basin of Λ. Let $\mathbf{x} \in W^s(\Lambda)$. Obviously, $\mathbf{f}^{[k]}(\mathbf{x}) \to \Lambda$ as $k \to \infty$. We recall that $U = \operatorname{int} \cdot A(L)$ is a fundamental neighborhood of Λ. Hence, there exists a $k_0 \in \mathbb{N}$, so that $\mathbf{f}^{[k]}(\mathbf{x}) \in A(L)$ for all $k > k_0$. Consider next the semi-trajectory $T^+(\mathbf{x}) = \{\mathbf{f}^{[k]}(\mathbf{x}) \,|\, k \in \mathbb{N}\}$. This semi-trajectory $T^+(\mathbf{x})$ generates an admissible path $\omega = \{c_{i_k} \mid \mathbf{f}^{[k]}(\mathbf{x}) \in M(i_k), k \in \mathbb{N}\}$, see also Theorem 1 in [Osi04]. Obviously, $c_{i_k} \in L$ for all $k > k_0$. Hence, $c_{i_0} \in D_u(L)$ and $\mathbf{x} \in M(i_0) \subseteq W_u(L)$. □

Theorem 5.6. *If L^s is an attractor on a symbolic image, so that $A(L^1) \supseteq A(L^2) \supseteq \cdots$ and that there is an attractor Λ, $\lim_{s\to\infty} A(L^s) = \Lambda$, then the basin $D_u(L^s)$ is such that the sequences $W_u(L^s)$ are embedded into each other*

$$W_u(L^1) \supseteq W_u(L^2) \supseteq \ldots \supseteq W^s(\Lambda).$$

Proof. According to Proposition 5.3 and the assumption $A(L^{\tilde{s}}) \supseteq \Lambda$, we conclude that $W_u(L^{\tilde{s}}) \supseteq W^s(\Lambda)$ for every $\tilde{s} \geq 0$. Let $C^{\tilde{s}}$ be a covering of M. A covering $C^{\tilde{s}+1}$ is a subdivision of $C^{\tilde{s}}$. Consequently, each cell $M(i)$ is subdivided so that new cells $m(i,k), k = 1,2,\ldots$, form a subdivision of the cell $M(i)$,

$$\bigcup_k m(i,k) = M(i)$$

Denote by $G^{\tilde{s}}$ and $G^{\tilde{s}+1}$ the symbolic images of $C^{\tilde{s}}$ and $C^{\tilde{s}+1}$. Then there is a natural mapping $h : G^{\tilde{s}+1} \mapsto G^{\tilde{s}}$, $h(c_{i,k}) = c_i$, which takes the graph $G^{\tilde{s}+1}$ onto $G^{\tilde{s}}$, i.e. each cell $c_{i,k}$ is mapped to the cell c_i, and the directed edge $c_{i,k} \to c_{j,l}$ is mapped to the directed edge $c_i \to c_j$. Hence, each path on $G^{\tilde{s}+1}$ is transformed to a path on $G^{\tilde{s}}$. Let $L^{\tilde{s}}$ and $L^{\tilde{s}+1}$ be the chosen attractors on $G^{\tilde{s}}$ and $G^{\tilde{s}+1}$ so that,

according to the assumption, $A(L^{\tilde{s}}) \supseteq A(L^{\tilde{s}+1})$. Obviously,

$$A(L^{\tilde{s}}) = \left\{ \bigcup M(i), c_i \in L^{\tilde{s}} \right\} \supseteq \left\{ \bigcup M(i), c_{i,k} \in L^{\tilde{s}+1} \right\} \supseteq \left\{ \bigcup m(i,k), c_{i,k} \in L^{\tilde{s}+1} \right\} = A(L^{\tilde{s}+1}).$$

We conclude that $L^{\tilde{s}} \supseteq h(L^{\tilde{s}+1})$. Hence, also $D_u(L^{\tilde{s}}) \supseteq h(D_u(L^{\tilde{s}+1}))$. It follows that,

$$W_u(L^{\tilde{s}}) = \left\{ \bigcup M(i), c_i \in D_u(L^{\tilde{s}}) \right\} \supseteq \left\{ \bigcup M(i), c_{i,k} \in D_u(L^{\tilde{s}+1}) \right\} \supseteq \left\{ \bigcup m(i,k), c_{i,k} \in D_u(L^{\tilde{s}+1}) \right\} = W_u(L^{\tilde{s}+1}).$$

□

We conclude that the upper bound of the basin, $W_u(L^s)$, shrinks while the lower bound $W(L^s)$ grows for $s \to \infty$. Furthermore, let us consider that for a subdivision cascade

$$W_u(L^1) \supseteq W_u(L^2) \supseteq \ldots \supseteq W^s(\Lambda) \supseteq \ldots \supseteq W(L^2) \supseteq W(L^1).$$

The upper bound $W_u(L^s)$ includes the real basin $W^s(\Lambda)$, while $W(L^s)$ is only a subset of it. More precisely, the borders of the area covered by the basin $W^s(\Lambda)$ lie in the additional cells of the upper bound. Due to Theorem 5.6, a sequence $W_u(L^s)$ is an outer covering for the solution $W^s(\Lambda)$ and, hence, can be computed by the multilevel subdivision scheme proposed in Sec. 3.1.

However, note that we did not explicitly prove that $\lim_{s\to\infty} W_u(L^s) = W^s(\Lambda)$ as $\delta(C^s) \to 0$. Reason for this is that we can compute the lower bound $W(L^s)$ for every $W_u(L^s)$, see Eq. 5.5 and Alg. 5.2. Due to the fact that $\lim_{s\to\infty} W(L^s) = W^s(\Lambda)$ as $\delta(C^s) \to 0$, see Theorem 5.2, we can then approximate $W^s(\Lambda)$ as precisely as we like. In practice, only for the last symbolic image, which will not be subdivided anymore, it is necessary to compute $D(L)$ and $W(L^s)$.

Let us recall that the whole area of interest must be selected for subdivision. According to Theorem 5.2, this would require to select in each subdivision stage s the attractor L^s, its domain of attraction $D_u(L^s)$ and the repeller L^{*s}. Consider now that, due to Proposition 5.2, $V(G^s) = D(L^s) \cup L^{*s}$. Hence, all cells of G^s would be subdivided. This is not desirable in a numerical computation, because the maximal size of a symbolic image is limited by the memory space. For this reason we propose a further modification of the algorithm. In each subdivision step, only the sets L^s and $D_u(L^s)$ but not L^{*s} are selected and subdivided. As a

result, the area $R(L^s)$ is discarded and will not be further investigated. We show now that the algorithm still works properly under these circumstances.

Let us consider the cell c_∞ and its corresponding area $M(\infty)$, see Def. 3.4. In case the area $R(L^{(s-1)})$ of the repeller $L^{*(s-1)}$ was not subdivided, then $L^{*s} = \{c_\infty\} \cup (D_u(L^s) \setminus D(L^s))$ because $L^{*s} = V(G^s) \setminus D(L^s)$, and, hence, $R(L^s) = M(\infty) \cup (W_u(L^s) \setminus W(L^s)) = \mathcal{M} \setminus W(L^s)$. The area $(W_u(L^s) \setminus W(L^s))$ can be considered as the edge between $R(L^s)$ and $W(L^s)$. We recall that if $\delta(C^s) \to 0$ as $s \to \infty$ then $\lim_{s\to\infty} W(L^s) = W^s(\Lambda)$. Consequently, $\lim_{s\to\infty} R(L^s) = \mathcal{M} \setminus W^s(\Lambda) = \Lambda^*$, see also Proposition 2.1. So we have seen that the explicit computation of $R(L^s)$ is not required in order to compute an approximation of the domain of attraction and of an attractor by symbolic analysis.

Remark 5.2. *It should be considered that c_∞ as denoted by Def. 3.4 does not have any outgoing edges $c_\infty \to c_i$. This is no restriction for our conclusions. Reason for this is that the only outgoing edge is $c_\infty \to c_\infty$. This edge is taken for granted. There can not be any other outgoing edges $c_\infty \to c_i$. We show this by contradiction. If such an outgoing edge exists then $c_i \in D_u(L^s)$ and, consequently, $c_\infty \in D_u(L^s)$. But this is a contradiction because $W_u(L^s)$ belongs to the area of investigation and, hence, $W_u(L^s) \cap M(\infty) = \emptyset$.*

Remark 5.3. *The proposed algorithm works fine in theory, i.e. for the case that the area of investigation is the domain $\mathcal{M}$ of $\mathbf{f} : \mathcal{M} \mapsto \mathcal{M}$. Unfortunately, in a practical implementation, the area of investigation M is only a part of $\mathcal{M}$, i.e. $M \subset \mathcal{M}$. Consequently, the domain of attraction $W^s(\Lambda)$ might partly lie outside of M so that $W^s(\Lambda) \bigcap (\mathcal{M} \setminus M) \neq \emptyset$. For such an environment, the proposed scheme does not work properly because, by definition, $(\mathcal{M} \setminus M) \subset M(\infty)$ for every $M(\infty)$, and we stated that $M(\infty) \subset \Lambda^*$. By our algorithm, we assume that every trajectory which leaves the area of investigation belongs to the repeller. But in case $W^s(\Lambda)$ lies partly outside of M, it might be that some trajectories leave M but then return. Up to now, this is still an open problem for our algorithm which causes numerical artifacts, see also the results of the numerical case studies in Sec. 5.7. However, the numerical artifacts does not affect the fact that $W(L^s)$ is a lower bound of $W^s(\Lambda)$. Every trajectory started in $W(L^s)$ ends in the attractor.*

5.2 Aspects of Filtration

In order to construct attractors and repellers by symbolic analysis, the concept of filtration is a helpful tool. We will give an overview about how this concept is connected with symbolic analysis, and how it can be used in order to construct

attractors and repellers. Hereby, the focus is put on the practical application of the method.

5.2.1 Filtrations on a Symbolic Image

We refer here to the theoretical concepts about filtration on symbolic images according to Osipenko [Osi99]. However, it is not our main intention to construct a fine filtration F as proposed in [Osi99]. We rather more focus on some aspects of these concepts which are important in practice and which we can also use later for the construction of attractors and repellers, see Sec. 5.3. Despite that, we will see that the algorithms introduced here are also sufficient tools to construct filtrations for a symbolic image G.

Definition 5.4. [Osi99] *A finite sequence $\phi = \{B_0, B_1, \ldots, B_m\}$ of cells on a symbolic image G is called a* filtration *on G if*

$$\emptyset = B_0 \subset B_1 \subset \cdots \subset B_m = V(G)$$

and if for each $B_k, k = 1,2,\ldots,m$ the condition holds that if a cell c_i of an edge $c_i \to c_j$ lies in B_k then also c_j lies in B_k.

The second condition means that there is no exit from B_k. Let L_k be a maximal invariant set in B_k.

Proposition 5.4. [Osi99] *Each maximal invariant set $L_k \subset B_k$ is an attractor and*

$$\emptyset = L_0 \subset L_1 \subset \cdots \subset L_m = V(G)$$

This proposition clarifies why a filtration can be used to construct attractors. Hereby, each set B_k corresponds to a fundamental neighborhood $U = \mathrm{int}\{\bigcup M(i), c_i \in B_k\}$ of an attractor.

We introduce a quasi-order relation on the cells of G. We set $c_i \prec c_j$ (or $c_j \succ c_i$) if and only if there exists a path from c_i to c_j,

$$c_i \to \cdots \to c_j.$$

Hence, a cell c_i is recurrent iff $c_i \prec c_i$, and recurrent cells c_i, c_j are equivalent if and only if $c_i \prec c_j \prec c_i$.

Definition 5.5. *The cells of a symbolic image G have an order $c_i < c_j$ (or $c_j > c_i$) if $c_i \prec c_j \not\prec c_i$. Cells are said to be equal, $c_i = c_j$, if $c_i \prec c_j \prec c_i$ or $i = j$.*

There is no order defined for cells c_i, c_j if $c_i \not\prec c_j$ and $c_j \not\prec c_i$. From the definition it follows that all cells $c_i \in L$, $c_j \in D(L)$ and $c_k \in L^*$ have an order $c_k < c_j < c_i$.

For a better understanding of the structural characteristics regarding a dynamical system, we are now interested in the classes

$$\mathcal{L}(H_p) = \{H_q \mid c_j > c_i,\, c_i \in H_p, c_j \in H_q\}, \tag{5.23}$$

which are denoted here as the *larger sets* of H_p. Also of interest are the connections between H_p and all of these sets,

$$\mathcal{CL}(H_p) = \{c_j \mid c_j \geq c_i,\, c_i \in H_p\}. \tag{5.24}$$

Furthermore, the connections between a set H_p and distinct sets $H_q \in \mathcal{L}(H_p)$,

$$\mathcal{CS}(H_p, H_q) = \{c_k \mid c_k \geq c_i \text{ and } c_j \geq c_k,\, c_i \in H_p,\, c_j \in H_q\}, \tag{5.25}$$

are required for our investigations. A cell belonging to $\mathcal{CS}(H_p, H_q)$ is denoted as a *connecting cell* between H_p and H_q. We can then localize the corresponding areas in the domain space:

$$U_{\mathcal{CL}}(H_p) = \left\{ \bigcup M(i) \mid c_i \in \mathcal{CL}(H_p) \right\}, \tag{5.26}$$

$$U_{\mathcal{CS}}(H_p, H_q) = \left\{ \bigcup M(j) \mid c_j \in \mathcal{CS}(H_p, H_q) \right\}. \tag{5.27}$$

We will provide algorithms to calculate $\mathcal{L}(H_p), U_{\mathcal{CL}}(H_p)$ and $U_{\mathcal{CS}}(H_p, H_q)$. The application of these algorithms allows the user to construct a filtration as described in the following.

The sets of equivalent recurrent cells $H_p \in \zeta$ with $p = 1, \ldots, |\zeta|$ can be renumbered so that if

$$\begin{aligned} &c_j > c_i \text{ with } c_i \in H_p, c_j \in H_q \\ &\quad \Rightarrow q > p \text{ and } q = 1, \ldots, |\zeta| \end{aligned} \tag{5.28}$$

Let us then define B_k:

$$B_k = \{c_j \mid c_j \geq c_i,\, c_i \in H_p,\, p = |\zeta| + 1 - k\}. \tag{5.29}$$

We could then construct a filtration as follows.

Proposition 5.5. [Osi99] *Let the symbolic image G be a connected graph. The finite sequence $\phi = \{\emptyset = B_0, B_1, \ldots, B_{|\zeta|+1} = V(G)\}$ is a fine filtration on G.*

We already stated that each B_k is a fundamental neighborhood of an attractor. We denote $U_k = \{\bigcup M(i) \,|c_i \in B_k\}$.

Proposition 5.6. [Osi99] *The sequence* $F = \{\emptyset = U_0, U_1, \ldots, U_{|\zeta|+1}\}$ *is a filtration for the dynamical system* **f**.

Obviously, we can get such a filtration F if we renumber the sets H_p of a symbolic image G according to Eq. 5.28. This can be achieved by analyzing their order relations $\mathcal{L}(H_p)$. Then, each $U_k, 0 < k < |\zeta|$, is described by the set $U_{C\mathcal{L}}(H_p)$ with $p = |\zeta| + 1 - k$.

5.2.2 Order Relations

Due to the fact that the dag graph DG is acyclic and directed, it can be said that for all cells $\tilde{c}_i, \tilde{c}_j \in DG$ there is either

1. $\tilde{c}_i > \tilde{c}_j$ or
2. $\tilde{c}_i < \tilde{c}_j$ or
3. no order relation defined for them ($\tilde{c}_j \not\succ \tilde{c}_i$ and $\tilde{c}_j \not\prec \tilde{c}_i$).

The same relations exist between the corresponding cells of $\tilde{c}_i$ on G. Additionally, if $\tilde{c}_i$ corresponds to a set H_k, so that $\tilde{c}_i \in DV(\zeta)$, then all cells in H_k are equal, $c_i = c_j$ with $c_i, c_j \in H_k$.

Consequently, a graph DG has already an internal representation of the larger sets $\mathcal{L}(H_k)$ for every H_k of G. It can easily be traversed to acquire a $\mathcal{L}(H_k)$. More precisely, a breadth-first search is started for the cell $\tilde{c}_k$ corresponding to H_k. Every cell $\tilde{c}_j$ with $\tilde{c}_j \in DV(\zeta)$ which is visited corresponds to a larger set H_j and will be marked. All marked sets H_j belong to $\mathcal{L}(H_k)$.

Proposition 5.7. *The algorithm as described above computes the larger sets* $\mathcal{L}(H_k)$ *for a* $H_k \in \zeta$.

Proof. If a breadth-first search is applied on a cell $\tilde{c}_k \in DV(\zeta)$ corresponding to a H_k then all cells $\tilde{c}_j \succ \tilde{c}_k$, and respectively $\tilde{c}_j > \tilde{c}_k$, are visited. Each of these $\tilde{c}_j$ corresponds either to a non-recurrent cell c_j (if $\tilde{c}_j \in DV(G)$), or to a recurrent cell set $H_j \in \zeta$ (if $\tilde{c}_j \in DV(\zeta)$). In the second case, H_j is a larger set because $\tilde{c}_j > \tilde{c}_k$ and so also

$$\forall c_i \in H_j, c_l \in H_k : c_i > c_l.$$

Hence, all sets visited during the traversal are larger than H_k. All of them get a mark.

No other sets are larger than H_k. We show this by contradiction. Imagine there is a set H_n with a corresponding cell $\tilde{c}_n$ which is larger than H_k but was not visited by the breadth-first search. The set H_n is larger than H_k if there is the relation $c_i > c_l$ for all $c_i \in H_n, c_l \in H_k$. To statisfy $c_i > c_l$ there must be a path from c_l to c_i and $H_k \neq H_n$. Hence, there must also be a path from $\tilde{c}_k$ to $\tilde{c}_n$. But this is a contradiction because if $\tilde{c}_n$ was not visited during the breadth-first search then there is also no path from $\tilde{c}_k$ to $\tilde{c}_n$.

Hence, after the traversal is finished, the marked sets are the larger sets $\mathcal{L}(H_k)$ for H_k. □

Note that we could apply a *topological sorting* on DG, see also [Sed93], to renumber the sets H_k so that they satisfy Eq. 5.28. We have not yet implemented such an algorithm because in practice this turned out to be of minor importance.

5.2.3 Connecting Recurrent Sets

Let us now discuss an implementation for $\mathcal{CL}(H_k)$. For this reason, we consider a selection $L = H_k$ and detect the upper bound of the *inverse* domain of attraction for L. We define

$$D_u^{-1}(L) = \{c_i \mid \text{there is a path through } c_i \text{ which starts in } L\} \tag{5.30}$$

as the *inverse upper bound of the basin*. We can locate $D_u^{-1}(L)$ on almost the same way than $D_u(L)$, see Alg. 5.1. A breadth-first search is applied on every $\tilde{c}_k \in DV(\zeta)$ corresponding to a $c_k \in L$. We traverse over all forward edges instead of the backward edges. The cell $\tilde{c}_k$ and every visited cell $\tilde{c}_j$ get marked. All cells in G corresponding to them belong to $D_u^{-1}(L)$. If $L = H_k$ then all the cells $c_j \in D_u^{-1}(L)$ form $\mathcal{CL}(H_k)$.

Proposition 5.8. *The algorithm described above calculates* $\mathcal{CL}(H_k)$.

Proof. The localisation of the inverse basin $D_u^{-1}(L)$ by breadth-first search is trivial and follows almost by definition. If we set $L = H_k$ and start a breadth-first search for $\tilde{c}_k \in DV(\zeta)$ corresponding to H_k we get all cells $\tilde{c}_j$ with paths

$$\tilde{c}_k \rightarrow \cdots \rightarrow \tilde{c}_j$$

Hence, there is the order relation $\tilde{c}_j > \tilde{c}_k$ defined for all these cells $\tilde{c}_j$ and also their corresponding cells on G. No other cells in G than the visited ones can have this order relation. Furthermore, there is the relation $c_i = c_j$ for all cells of the set

H_k corresponding to $\tilde{c}_k$. So the cells corresponding to the marked cells $\tilde{c}_j$ on DG form the set with all $c_j \geq c_m, c_m \in H_k$ and we conclude that

$$\mathcal{CL}(H_k) = D_u^{-1}(H_k).$$

□

We focus now on the computation of the set $\mathcal{CS}(H_p, H_q)$. This set can be acquired by the combination of two algorithms. First we compute $\mathcal{CL}(H_p)$. We give every visited cell $\tilde{c}_i$ a special mark $\diamond$. Afterwards, we calculate $D_u(H_q)$. For every visited cell $\tilde{c}_j$ we check if it has a mark $\diamond$. If so, it gets marked as a being a connecting cell between H_p and H_q. Every cell on G corresponding to such a connecting cell belongs to $\mathcal{CS}(H_p, H_q)$.

Proposition 5.9. *The algorithm described above locates all cells belonging to* $\mathcal{CS}(H_p, H_q)$.

Proof. We assume $\tilde{c}_p$ as the cell on DG corresponding to H_p, and $\tilde{c}_q$ as the one corresponding to H_q. If $\mathcal{CL}(H_p)$ is computed and the cells which are visited get a mark $\diamond$, then every cell $\tilde{c}_i \geq \tilde{c}_p$ is marked with $\diamond$. If then $D_u(H_q)$ is computed, we visit all cells $\tilde{c}_j \leq \tilde{c}_q$. For each of these cells $\tilde{c}_j$ we can conclude that $\tilde{c}_j \geq \tilde{c}_p$ if $\tilde{c}_j$ has a mark $\diamond$. Consequently, the cells $\tilde{c}_j$ with a mark $\diamond$ and visited by $D_u(H_q)$ are the connecting cells between H_p and H_q because $\tilde{c}_j \geq \tilde{c}_p$ and $\tilde{c}_q \geq \tilde{c}_j$. The cells on G corresponding to them are the cells belonging to $\mathcal{CS}(H_p, H_q)$. □

The cells belonging to $\mathcal{CS}(H_p, H_q)$ can be considered as an upper bound. Let us then define the lower bound on DG as

$$\begin{aligned} \mathcal{CS}_l(H_p, H_q) = \{c_i \mid &\text{each path through } d(c_i) \\ &\text{goes to } \tilde{c}_q \wedge c_i \in \mathcal{CS}(H_p, H_q)\}. \end{aligned} \tag{5.31}$$

This lower bound can be acquired almost on the same way than $\tilde{D}(L)$, see Alg. 5.2. First we apply the algorithm for $\mathcal{CS}(H_p, H_q)$. Then we start a reverse breadth-first search for every cell $\tilde{c}_l$ which was not marked. Every cell $\tilde{c}_i$ visited by traversing over the backward edges of each $\tilde{c}_l$ has at least one path which does not lead to $\tilde{c}_q$ and, hence, it can not belong to the lower bound. If it is marked, it will be unmarked. All remaining cells with a mark belong to the lower bound.

In practice, the user should use the same strategy to acquire the lower bound as proposed for the localization of the basin. The upper bound, $\mathcal{CS}(H_p, H_q)$, is located for every symbolic image which will be subdivided. Only for the final image, which will not be subdivided anymore, should the lower bound $\mathcal{CS}(H_p, H_q)$ be detected.

5.3 Construction of Attractors and Repellers

Our next task is the study of methods to construct attractors and repellers. In Sec. 5.1.3 we introduced user-defined selections S_H, see Eqs. 5.14 and 5.15, in order to pick invariant sets and some attractors. We will now present methods to compute attractors and repellers in general from such selections S_H. These methods are built upon those used to construct filtrations. Indeed, the construction scheme we propose here is closely related with filtrations, and requires the calculation of order relations and connecting cells.

We consider also some restrictions due to the limitations of numerical symbolic image construction which were neglected in the theoretical studies of Osipenko. For a homeomorphism $\mathbf{f}$, which is defined on a manifold $\mathcal{M}$, one can usually only investigate a limited area $M \subset \mathcal{M}$. Hence, some of the trajectories may leave the area M investigated by symbolic analysis, and others may come from $\mathcal{M} \setminus M$. In that case, we can not avoid that attractors which are only partly covered by M can not be found. However, it is our intention to verify at least that each computed outer covering of an attractor or repeller really contains such an object. Therefore, we must consider the incoming and outgoing trajectories of a covering. In [KMO03] these problems were already discussed for invariant sets. We refer to these results and introduce the following, extended definitions.

Definition 5.6. *The cell c_i of G is called* incoming *if it has no entry edge,*

$$\nexists c_j \in V(G) : c_j \to c_i.$$

If a cell c_i has no entry edge then the trajectories which pass $M(i)$ can neither come from another box $M(j)$ nor originate in $M(i)$ (if so, there would be an edge $c_i \to c_i$). Consequently, they originate outside of the covering C. Obviously, all such incoming cells can not be recurrent and, hence, are non-recurrent.

Definition 5.7. *A cell c_i is called* entering *if $c_j \leq c_i$ and c_j is an incoming cell.*

Definition 5.8. *A cell c_i of a symbolic image is called* leaving *if $c_i \leq c_\infty$.*

If c_i is a leaving cell it means that at least one trajectory passing through the corresponding box $M(i)$ enters $M(\infty)$. Hence, such a trajectory leaves the area covered by C. If c_i is an entering cell it means that at least one trajectory entering the corresponding box $M(i)$ comes from $M(\infty)$. Hence, such a trajectory enters the area covered by C.

Let us denote $O^+(G) \subseteq V(G)$ as the set of all leaving cells and $O^-(G) \subseteq V(G)$ as the set of all entering cells. We can assume that in a numerically computed

symbolic image G there is usually $O^+(G) \neq \emptyset$ and/or $O^-(G) \neq \emptyset$. Furthermore, we expect that the area of investigation $M \subset \mathcal{M}$.

Taking this into account, we will now discuss how to construct an attractor L which lies completely in our area of investigation, i.e. $c_\infty \notin L$. Let us recall that the user can select subsets $S_H \subseteq \zeta$, see Sec. 5.1.3. If for every $H_k \in S_H$ there is $Ex(H_k) = \emptyset$ then

$$L = \bigcup_{S_H} H_k$$

is an attractor because $Ex(L) = \emptyset$. Let us now assume we have a selection S_H with $Ex(H_l) \neq \emptyset$ for some sets $H_l \in S_H$. We can still construct an attractor for S_H if there exist sets $H_k \in S_H$ with $Ex(H_k) = \emptyset$ and

$$\forall c_i \in H_l, Ex(H_l) \neq \emptyset : c_i < c_j, c_j \in H_k, Ex(H_k) = \emptyset.$$

To show this, we define the following selections $S_k \subseteq \zeta$ which can be built recursively starting with S_1:

$$\begin{aligned} S_1 =& S_H,\, S_H \subseteq \{H_l \,|\, H_l \in \zeta,\, Ex(H_l) = \emptyset\}, \\ S_k =& \{H_m \,|\, \mathcal{L}(H_m) \subseteq S_{k-1} \text{ and } \mathcal{L}(H_m) \neq \emptyset\} \cup S_{k-1} \text{ for } k = 2, \ldots |\zeta|. \end{aligned} \tag{5.32}$$

Let us denote for a selection S_k the set $\mathcal{L}(S_k) = \bigcup_{H_j \in S_k} \mathcal{L}(H_j)$.

Lemma 5.1. *If there is a path ω through a cell c_i which belongs to a set $H_l \in S_k$ then for all following cells $c_j \geq c_i$ in ω we can state that c_j does not belong to a set $H_n \notin S_k$.*

Proof. If $k = 1$ this is trivial because the sets $H_l \in S_1$ have no exit edges. If $k > 1$ then consider that $S_{k-1} = \mathcal{L}(S_k)$ and $S_{k-1} \subseteq S_k$. For $k > 1$ we prove the Lemma by contradiction. Imagine there is a path ω leaving a set $H_l \in S_k$ which goes to a set $H_n \notin S_k$. Then H_n must be a larger set of H_l. But this is a contradiction because the sets $\mathcal{L}(H_l) \subseteq \mathcal{L}(S_k)$ belong to S_k. Hence, no such path exists. □

Theorem 5.7. *The set*

$$L(S_k) = \{c_i \,|\, c_i \in \mathcal{CS}(H_p, H_q) \text{ with } H_p, H_q \in S_k\}$$

is an attractor of G if $O^+(G) \cap L(S_k) = \emptyset$.

Proof. First we show that $L(S_k)$ is an invariant set. Consider that every $c_i \in L(S_k)$ belongs to a set $\mathcal{CS}(H_p, H_q)$. Consequently, if c_i is recurrent, it belongs to an equivalent recurrent cell set $H_q \in S_k$. If c_i is non-recurrent then it is on a path between two recurrent cell sets because there exist $H_p, H_q \in S_k$ so that

$$\forall c_p \in H_p \wedge \forall c_q \in H_q : c_p \leq c_i \leq c_q.$$

Hence, there is an admissible path passing through each $c_i \in L(S_k)$.

Next, we prove that $Ex(L(S_k)) = \emptyset$ by contradiction. Assume there is an exit for $L(S_k)$. Then there exists a finite path ω started in a recurrent cell $c_i \in H_p \in S_k$ which leaves $L(S_k)$. If this path finishes in a non-recurrent cell it will be extended until a recurrent cell or c_∞ is reached. This is always possible because every non-recurrent cell except c_∞ has at least one exit edge. Such a path can not finish in a recurrent cell set H_q belonging to S_k because all cells $c_i \in \mathcal{CS}(H_p, H_q)$ belong to $L(S_k)$. Hence, this path either finishes in c_∞ or in a recurrent cell set $H_n \notin S_k$. The condition $O^+(G) \cap L(S_k) = \emptyset$ excludes the first possibility. So ω must lead to a $H_n \notin S_k$. But, according to Lemma 5.1, this is also a contradiction. Hence, $Ex(L(S_k)) = \emptyset$ and $L(S_k)$ is an attractor. □

Corollary 5.1. *Note that $L(S_k)$ can also be defined as*

$$L(S_k) = \{c_i \,|\, c_i \in \mathcal{CL}(H_p) \text{ with } H_p \in S_k\}$$

because for every $H_p \in S_k$ all larger sets $\mathcal{L}(H_p)$ also belong to S_k. Hence, if the condition $O^+(G) \cap L(S_k) = \emptyset$ is fulfilled, we can say that

$$\bigcup_{S_k \times S_k} \mathcal{CS}(H_p, H_q) = \bigcup_{H_p \in S_k} \Big(\bigcup_{H_q \in \mathcal{L}(H_p)} \mathcal{CS}(H_p, H_q) \Big) = \bigcup_{H_p \in S_k} \mathcal{CL}(H_p).$$

A repeller L^* can be constructed by the same scheme. We define the selections $\tilde{S}_k \subseteq \zeta$ as

$$\begin{aligned} \tilde{S}_1 =& \tilde{S}_H \subseteq \{H_l \,|\, En(H_l) = \emptyset\}, \\ \tilde{S}_k =& \left\{H_m \,|\, H_m \in \mathcal{L}(\tilde{S}_{k-1}) \text{ and } H_m \notin \mathcal{L}(\zeta \setminus \tilde{S}_{k-1})\right\} \cup S_{k-1} \text{ for } k = 2, \ldots |\zeta|. \end{aligned} \tag{5.33}$$

Lemma 5.2. *If there is a path ω through a cell c_i which belongs to a set $H_l \in \tilde{S}_k$ then for all preceding cells $c_j \leq c_i$ in ω we can state that c_j does not belong to a set $H_n \notin \tilde{S}_k$.*

Proof. If $k = 1$ this is trivial because the sets $H_l \in \tilde{S}_1$ have no entry edges. If $k > 1$ then consider that for all $1 < k' \leq k$:

$$H_l \in \tilde{S}_{k'} \setminus \tilde{S}_{k'-1} : H_l \in \mathcal{L}(\tilde{S}_{k'-1}) \text{ and } H_l \notin \mathcal{L}(\zeta \setminus \tilde{S}_{k'-1})$$

and $\tilde{S}_{k'-1} \subseteq \tilde{S}_{k'}$. For $k > 1$ we prove the Lemma by contradiction. Imagine there is a path ω from a set $H_n \notin \tilde{S}_k$ which goes to a set $H_l \in \tilde{S}_k$. Then H_l must be a larger set of H_n. Furthermore, there is a $1 < k' \leq k$ so that $H_l \in \tilde{S}_{k'} \setminus \tilde{S}_{k'-1}$. But this is a contradiction because if $H_l \in \mathcal{L}(H_n)$ then H_n must belong to $\tilde{S}_{k'-1} \subseteq \tilde{S}_{k'}$ and so also to $\tilde{S}_k$. Hence, no such path exists. □

Theorem 5.8. *The set*

$$L^*(\tilde{S}_k) = \{c_i \,|\, c_i \in \mathcal{CS}(H_p, H_q) \text{ with } H_p, H_q \in \tilde{S}_k\}$$

is a repeller of G if $O^-(G) \cap L^*(\tilde{S}_k) = \emptyset$.

Proof. $L^*(\tilde{S}_k)$ is an invariant set, see proof to Theorem 5.7. So we only need to prove that $En(L^*(\tilde{S}_k)) = \emptyset$. We do this by contradiction. Assume there is an entry to $L^*(\tilde{S}_k)$. Then there exists a finite path ω which starts either in an incoming cell or a recurrent cell, then enters $L^*(\tilde{S}_k)$ and ends in a recurrent cell $c_i \in H_q \in S_k$. Such a path can not start in a recurrent cell set H_p belonging to $\tilde{S}_k$ because all cells $c_i \in \mathcal{CS}(H_p, H_q)$ belong to $L^*(\tilde{S}_k)$. Hence, such a path either starts in an incoming cell c_i or in a recurrent cell set $H_n \notin \tilde{S}_k$. The condition $O^-(G) \cap L^*(\tilde{S}_k) = \emptyset$ excludes the first possibility. So ω must originate in a $H_n \notin \tilde{S}_k$. But, according to Lemma 5.2, this is also a contradiction. Hence, $En(L^*(\tilde{S}_k)) = \emptyset$ and $L^*(\tilde{S}_k)$ is a repeller. □

Corollary 5.2. *Note that* $L^*(\tilde{S}_k)$ *can also be defined as*

$$L^*(\tilde{S}_k) = \{c_i \,|\, c_i \in D_u(L^s)\}, L^s = \bigcup_{\tilde{S}_k} H_q$$

because every $H_q \in \tilde{S}_k$ *has either no entrance or all* $c_j \leq c_i$ *with* $c_i \in H_q$ *belong to* $L^*(\tilde{S}_k)$. *Hence, if the condition* $O^-(G) \cap L(\tilde{S}_k) = \emptyset$ *is fulfilled, we can say that*

$$\bigcup_{\tilde{S}_k \times \tilde{S}_k} \mathcal{CS}(H_p, H_q) = \bigcup_{H_p \in \tilde{S}_k} \Big(\bigcup_{H_q \in \mathcal{L}(H_p) \cap \tilde{S}_k} \mathcal{CS}(H_p, H_q) \Big) = D_u(L^s).$$

We will now focus on the conditions $O^+(G) \cap L(S_k) = \emptyset$ and $O^-(G) \cap L^*(\tilde{S}_k) = \emptyset$ in order to develop algorithms which allow us to check these conditions. Let us define the sets

$$O^+(\zeta) = \left\{H_k | \, \exists c_i \in H_k : c_i \in O^+(G)\right\}, \tag{5.34}$$

$$O^-(\zeta) = \left\{H_k | \, \exists c_i \in H_k : c_i \in O^-(G)\right\}. \tag{5.35}$$

Then we can can conclude that

Lemma 5.3.

$$O^+(\zeta) \cap S_k = \emptyset \Leftrightarrow O^+(G) \cap L(S_k) = \emptyset.$$

Proof. If $O^+(G) \cap L(S_k) = \emptyset$ then also $O^+(\zeta) \cap S_k = \emptyset$ because all cells c_i belonging to the sets $H_l \in S_k$ also belong to $L(S_k)$.

If $O^+(\zeta) \cap S_k = \emptyset$ then also $O^+(G) \cap L(S_k) = \emptyset$. We prove this by contradiction. Let us assume that $O^+(\zeta) \cap S_k = \emptyset$ and there is a leaving cell c_l in $L(S_k)$. Obviously, c_l is not recurrent because if it would be recurrent then it would belong to a set $H_m \in S_k$. This would contradict $O^+(\zeta) \cap S_k = \emptyset$. Hence, c_l must be non-recurrent. If so, there exists an admissible path ω through c_l because $L(S_k)$ is an invariant set. Consequently, there is a cell $c_i \in L(S_k)$ in ω so that $c_i < c_l$ and c_i is recurrent. Obviously, c_i belongs to a $H_l \in S_k$ and is leaving. But this is a contradiction and, consequently, c_l can not be a leaving cell. □

Lemma 5.4.

$$O^-(\zeta) \cap \tilde{S}_k = \emptyset \Leftrightarrow O^-(G) \cap L^*(\tilde{S}_k) = \emptyset.$$

Proof. If $O^-(G) \cap L^*(\tilde{S}_k) = \emptyset$ then also $O^-(\zeta) \cap \tilde{S}_k = \emptyset$ because all cells c_i belonging to the sets $H_l \in \tilde{S}_k$ also belong to $L^*(\tilde{S}_k)$.

If $O^-(\zeta) \cap \tilde{S}_k = \emptyset$ then also $O^-(G) \cap L^*(\tilde{S}_k) = \emptyset$. We prove this by contradiction. Let us assume that $O^-(\zeta) \cap \tilde{S}_k = \emptyset$ and there is an entering cell c_l in $L^*(\tilde{S}_k)$. Obviously, c_l is not recurrent because if it would be recurrent then it would belong to a set $H_m \in \tilde{S}_k$. This would contradict $O^-(\zeta) \cap \tilde{S}_k = \emptyset$. Hence, c_l must be non-recurrent. If so, there exists an admissible path ω through c_l because $L^*(\tilde{S}_k)$ is an invariant set. Consequently, there is a cell $c_i \in L(S_k)$ in ω so that $c_i > c_l$ and c_i is recurrent. Obviously, c_i belongs to a $H_l \in \tilde{S}_k$ and is entering. But this is a contradiction and, consequently, c_l can not be an entering cell. □

It is now our intention to calculate $O^+(\zeta)$ and $O^-(\zeta)$. We can do this by modification of the algorithms for calculation of $D_u(L)$ and $D_u^{-1}(L)$ (see Eq. 5.30 and Theorem 5.3 and Proposition 5.8). We define a set $L^+ = \{\tilde{c}_\infty\}$, where $\tilde{c}_\infty$ corresponds to c_∞. Then we locate all cells $\tilde{c}_l \in DV(G)$ corresponding to non-recurrent cells. If $|P(\tilde{c}_l)| = 0$ we add $\tilde{c}_l$ to L^-. Afterwards, the upper bound of the basin is calculated for L^+ as well as the upper bound of the inverse basin for L^-. All cells $\tilde{c}_i$ belonging to $D_u(L^+)$ get a marking $\circ+$. The ones belonging to $D_u^{-1}(L^-)$ get a marking $\circ-$. The sets H_k corresponding to cells $\tilde{c}_k \in DV(\zeta)$ with a mark $\circ+$ belong to $O^+(\zeta)$, the ones corresponding to $\tilde{c}_k$'s with $\circ-$ belong to $O^-(\zeta)$.

Proposition 5.10. *The algorithm described above finds the sets $O^+(\zeta)$ and $O^-(\zeta)$ for G.*

Proof. We consider all incoming cells and c_∞ as invariant sets. Hence, the set L^+ in DG can be called an attractor because $Ex(L^+) = \emptyset$, and the set of incoming cells is said to be a repeller L^- because $En(L^-) = \emptyset$. We state now that, firstly, cells are leaving iff they have a path to $\tilde{c}_\infty$. Hence, all cells belonging to the upper bound of the basin of L^+ are leaving. Secondly, cells in DG are entering cells iff there is a path through it which starts in an incoming cell. Hence, all cells

belonging to the upper bound of the inverse basin of L^- are entering cells in case L^- is a set consisting of all incoming cells. If our algorithms for $D_u(L^+)$ and $D_u^{-1}(L^-)$ get applied then the cells in DG belonging to the basin $D_u(L^+)$ get a mark $\circ+$, the ones which belong to the inverse basin $D_u^{-1}(L^-)$ get $\circ-$. All sets corresponding to cells $\tilde{c}_k \in DV(\zeta)$ with a mark $\circ+$ are considered to be members of $O^+(\zeta)$, and the sets corresponding to cells $\tilde{c}_k \in DV(\zeta)$ with a mark $\circ-$ are considered to be members of $O^-(\zeta)$.

We conclude that our algorithm calculates $O^+(\zeta)$ and $O^-(\zeta)$ correctly if the condition is fulfilled that $L^+ = \{\tilde{c}_\infty\}$ and L^- is a set consisting of all incoming cells. Obviously, $L^+ = \{\tilde{c}_\infty\}$ is true. Secondly, the cells $\tilde{c}_l$ with $\tilde{c}_l \in DV(G)$ and $|P(\tilde{c}_l)| = 0$ are the cells with no entrances corresponding to non-recurrent cells in G. These are the incoming cells in DG. All of them and only they belong to L^-. □

The user can now use the algorithms described above to identify attractors L, repellers L^* and basins $D(L)$ in a symbolic image G. Usually he will perform some or all of the following steps:

1. Selection of a set $S_1 = S_H \subseteq \{H_k \,|\, Ex(H_k) = \emptyset, H_k \in \zeta\}$ (see Eq. 5.32). The list of elements $H_k \in \zeta$, which is needed to do so, can be computed for G. The information if $Ex(H_k) = \emptyset$ is also available because after creation of DG, a mark can be set for H_k if for the corresponding cell $\tilde{c}_k$ no target edges exist, i.e. $|E(\tilde{c}_k)| = 0$, and, hence, $Ex(H_k) = \emptyset$.

2. Recursive construction of a selection S_k, see Eq. 5.32. The list of larger sets $\mathcal{L}(H_l)$ for every $H_l \in \zeta$ is all that is need to do so. This list can also be computed, see Proposition 5.7.

3. Check if $L(S_k)$ is an attractor, see Theorem 5.7. Hence, one must check the condition described in Lemma 5.3 for every $H_l \in S_k$ and test that $H_l \notin O^+(G)$. The list $O^+(G)$ can be computed, see Proposition 5.10).

4. If $L(S_k)$ is an attractor, we can either construct this attractor by calculation of the connection cells $\mathcal{CL}(H_p)$ for every $H_p \in S_k$ (see Theorem 5.7 and Corollary 5.1) and/or we can construct the basin for $L = \bigcup_{S_k} H_l$. We might either choose to calculate the upper bound $D_u(L)$ or the lower bound $D(L)$, see Theorems 5.3, 5.4 and 5.5 – depending on the decision if the current symbolic image will be subdivided or not. Note also that $D(L) = D(L(S_k))$.

5. Selection of a set $\tilde{S}_1 = \tilde{S}_H \subseteq \{H_k \,|\, En(H_k) = \emptyset, H_k \in \zeta\}$, recursive construction of a desired $\tilde{S}_k$, see Eq. 5.33, and the test if $L^*(\tilde{S}_k)$ is a repeller by checking for all $H_l \in \tilde{S}_k$ the condition that $H_l \notin O^-(G)$ as stated in Lemma 5.4. The list $O^-(G)$ can be computed, see Proposition 5.10.

6. If $L^*(\tilde{S}_k)$ is a repeller then we can construct it by calculation of the upper bound of the basin $D_u(L)$ with $L = \cup_{\tilde{S}_k} H_k$ (see Theorem 5.8 and Corollary 5.2).

7. Subdivision of the symbolic image G^s into G^{s+1}. We subdivide the areas of interests, that is either the attractor $L(S_k)$, the upper bound of its basin $D_u(L(S_k))$, or the repeller $L^*(\tilde{S}_k)$. Note that after a subdivision only the subdivided type of objects can be detected. If, for instance, the user decides to subdivide an attractor in G^0 then he can approximate attractors by all following symbolic images $G^1, G^2, \ldots$ but no repellers or basins.

The selection of attractors and repellers is performed by the user. The data he needs to do so can be acquired by our algorithms. However, proper selections for S_k as well as the interpretation of the computed results is still the duty of the user.

5.4 Detecting (Un)stable and Connecting Manifolds

The algorithms we introduced so far for the localization of attractors, repellers and the domain of attraction can also be used for other tasks. In this section we show further applications. Hereby, we avoid complete proofs of correctness. Reason for this is that they might be extensive if given in detail. Instead, the rough outline we present here should be sufficient to get the essentials and make it obvious to the reader that the proposed methods can be applied properly. In Sec. 5.7 we give also numerical examples for the techniques.

We first look at the stable manifold. Let us recall that its definition is similar to those of the domain of attraction, compare Eqs. 2.8 and 2.12. Indeed, the difference between them is the target. The basin of attraction is the union of points which are attracted by an attractor, and the stable manifold is, in our definition, the union of points attracted by a saddle. In our context, we restrict ourselves to fixed points or periodic orbits of saddle type as targets. Let $P = \{\mathbf{x}_0, \ldots, \mathbf{x}_p\}$ be a periodic orbit of saddle type. Note that such a saddle is, of course, chain recurrent. Consequently, using the introduced methods of symbolic analysis, this object can be approximated by an outer covering, see Sec. 3.3. In the corresponding symbolic image graph G it is represented by a recurrent set $H_k \subseteq \zeta$. If $S_H = \{H_k\}$ is now a selection as described in Sec. 5.1.3 then the upper bound of the basin on G, $D_u(H_k)$, can be computed, see Sec. 5.1.4. The area $W_u(H_k)$ is an approximation of the stable manifold $W^s(P)$. Note that we have to consider that $L(H_k)$ has an exit, $Ex(L(H_k)) \neq \emptyset$, because it is not an attractor. However, this is of no importance for the computations, as already mentioned in Sec. 5.1.3.

In order to select the proper set H_k for S_H as target, it is necessary to decide, if the area corresponding to H_k is an outer covering of a saddle. Up to now, this can not be done directly on G as in the case of an attractor. However, a possible approach could be that the saddle is first localized in a different computation by the application of basic investigation methods, see Sec 3.3, and an analysis of the Jacobian matrices of the solutions. If a saddle has been found then a symbolic image graph can be constructed for the computation of the stable manifold. In such a graph, those H_k is selected as target whose corresponding area covers the neighborhood of the formerly computed saddle points.

Similar to the stable, also the unstable manifold can be approximated. Let us therefore consider that the unstable manifold is the sibling of the stable one. More precisely, if $W^s(P)$ is the stable manifold for $\mathbf{f}$ then it is the unstable manifold for $\mathbf{f}^{[-1]}$ and vice versa, compare Eqs. 2.9 and 2.10. This lets us conclude that if $W^s(P)$ can be approximated on the symbolic image G by computation of the basin, then $W^u(P)$ can be approximated by the inverse basin, namely the upper bound $D_u^{-1}(L)$. Note that hereby it is of no importance if the inverse $\mathbf{f}^{[-1]}$ exists and can be computed because the inverse basin is computed on G, and G can be constructed without the computation of $\mathbf{f}^{[-1]}$.

Another field of application for our algorithms is the computation of connecting manifolds, see Definition 2.11. The connecting manifold was defined as the intersection $W^u(Q_i) \cap W^s(Q_j)$ of two components Q_i, Q_j of the chain recurrent set. Such a connecting manifold can be approximated by the computation of the intersection of a basin and an inverse basin of two recurrent sets $H_i, H_j \in \zeta$. This can be done by computation of the connections $\mathcal{CS}(H_i, H_j)$ between H_i and H_j, see also Proposition 5.9. In the last section, we mentioned connecting manifolds in the context of the construction of attractors and repellers. However, the concept can also be applied in a more general approach. For instance, also the connecting manifolds between saddles can be localized.

5.5 Performance Analysis

The performance of the algorithms described in this chapter is analyzed by studying the worst-case scenario. We refer here to the results of Sec. 3.4, especially Proposition 3.2. Let us recall that the performance time for the construction of G, the localization of the recurrent cells $RV(G)$ and the sets ζ is in $O(n \cdot \log(n))$ and depends on the number of boxes in the covering C because $|C| = |V(G)| = n$. Furthermore, the number of edges per cell $c_i \in G$, denoted here by $e(c_i)$, is limited by a constant k, so that $e(c_i) \leq k$. This is due to the fact that a user-defined number

of k scan points per box $M(i)$ are used for the computation of the edges.

Remark 5.4. *Localization of $D_u(L)$, $\tilde{D}(L)$ and $D(L)$, see Theorems 5.3, 5.4 and 5.5, is in $O(n)$.*

Localization of the domain of attraction for L requires it to perform a reversed breadth-first search for each $\tilde{c}_i \in L$. Note that the search can always be stopped at a cell $\tilde{c}_j$ which was visited before because then all cells $\tilde{c}_k > \tilde{c}_j$ were also visited and investigated before, see Algorithm 5.1. So every edge of DG is traversed only once, no matter how large L is, and the search algorithm is in

$$O(|e(DG)|) \subseteq O(k \cdot n) \subseteq O(n),$$

where $|e(DG)|$ is the number of edges in DG. In terms of performance, the algorithm to detect the lower bound $D_u(L)$ behaves exactly the same. The only difference is that it is started from the cells $\tilde{c}_k$ which do not belong to the upper bound of the domain of attraction, i.e. $\tilde{c}_k \in V(DG) \setminus D_u(L)$. So it is also in $O(n)$.

The extensions of the algorithm to get $D(L)$ only reduce the number of cells in $D_u(L)$ by unmarking all cells $\tilde{c}_u \in DV(\zeta) \setminus L$. This operation is in linear time. Hence, the detection of $D(L)$ is also in $O(n)$.

Remark 5.5. *Detection of the larger sets $\mathcal{L}(H_k)$ for a $H_k \in \zeta$, see Proposition 5.7 is in $O(n)$. Detection of larger sets for all $H_k \in \zeta$ is in $O(|\zeta| \cdot n) \subseteq O(n^2)$.*

A breadth-first search is started on the cell $\tilde{c}_k$ corresponding to H_k. For each visited cell $\tilde{c}_j$ we check in $O(1)$ if $\tilde{c}_j \in \zeta$ and, if so, add the corresponding set H_j to the list $\mathcal{L}(H_k)$. During the search every edge of DG might get visited but every edge is visited not more than once. It follows that the algorithm is in

$$O(e(DG) \cdot 1) \subseteq O(k \cdot n) \subseteq O(n).$$

Usually we have to detect the larger sets for every set $H_k \in \zeta$. So we have to start our algorithm for each set H_k. This takes time

$$O(|\zeta| \cdot n) \subseteq O(n^2).$$

Remark 5.6. *Detection of $\mathcal{CL}(H_k)$, see Proposition 5.8, is in $O(n)$.*

The detection of $\mathcal{CL}(H_k)$ is achieved by localization of the inverse domain of attraction. We use almost the same algorithm than for detection of the basin and perform again a breadth-first search. Hence, this algorithm is in $O(n)$.

Remark 5.7. *Connection of selected cell sets $\mathcal{CS}(H_p, H_q)$, see Proposition 5.9, is in $O(n)$.*

To acquire $\mathcal{CS}(H_p, H_q)$, we calculate $\mathcal{CL}(H_p)$ and $D_u(H_q)$. According to remarks 5.6 and 5.4 both operations can be performed in linear time. Hence, the algorithm is in $O(n)$.

Remark 5.8. *Calculation of the sets $O^+(\zeta)$ and $O^-(\zeta)$, see Proposition 5.10, is in $O(n)$.*

First thing to do is the selection of the sets L^+ and L^-. A traverse over all cells $\tilde{c}_l \in DV(G)$ is sufficient to check the necessary conditions and get the lists of elements. Afterwards, we detect $D_u(L^+)$ and $D_u^{-1}(L^-)$ and set appropriate markings. According to Theorem 5.4 this operation can be performed in linear time. Hence, the algorithm is in

$$O(|DV(G)| + n) \subseteq O(n).$$

It turns out that most operations on attractors and their basins do not exceed linear computation time. Only the calculation of the larger sets might be in $O(n^2)$, see Remark 5.5. But it should be mentioned that, in general, the detection of larger sets is only useful if the number $|\zeta|$ is small. In that case, the algorithm does not really slow down performance.

So the crucial aspect of a computation is the construction of the symbolic image and not the investigations performed on it. In Chapter 3 an implementation was introduced which constructs symbolic images in time $O(n \cdot \log(n))$, see Proposition 3.2. Hence, our implementation works efficiently. Note, however, that an investigation based on the methods proposed in this chapter is in general rather more limited by memory space than by performance time.

5.6 Comparison with Other Approaches

We compare now the introduced investigation methods with those of others. The first technique to look at is the computation of the domain of attraction. In Sec. 2.4 we have already mentioned an alternative method which is based on forward iteration. We have pointed out that there are two main problems. First of all, the number of forward iterates applied to a trajectory is limited. Second, the transient and the asymptotic phase might be difficult to distinguish. Both problems do not exist in our approach. The domain of attraction is localized by the breadth-first search, so that each cell is only visited once, no matter how many forward iterates a trajectory started in this cell requires to reach the asymptotic state. Furthermore, the areas of asymptotic state are approximated by the outer

covering of the attractor. Consequently, the distinction between transient and asymptotic phase is clear. Note that the accuracy of this outer covering can be as precise as one likes, at least theoretically.

A different implementation, which combines phase space discretization with forward iterates, was proposed by [NY97]. Hereby, the area of investigation is divided into boxes, and a forward iterate started for each box. The software keeps track of the results of former forward iterates. If a forward iterate passes a box which has already been visited then the former results can be taken as a solution for the current trajectory as well. Although this means that both above mentioned problems of computation can be solved, the accuracy is quite low. More precisely, the state of each box is only determined by the results of one trajectory passing through it. In contrast, by our technique several trajectories within a box, i.e. the scan points, are investigated. For this reason, it can be decided if an area belongs to the upper or lower bound of a basin. Even if only a limited number of boxes can be computed, depending on the size of the memory space, the distinction if such a box belongs to the upper or lower bound can be performed to an accuracy which depends mainly on the required computation time, i.e. the number of scan points used for the construction of the symbolic image graph. Besides that, a further advantage of our approach is the application of multilevel subdivision in order to achieve more precise results, and the fact that the attractor(s) for which the basin should be computed, can be chosen by the user.

A further technique for the computation of the domain of attraction was proposed by Hsu [Hsu87]. The approach is based on cell mapping. Hence, it can be considered close to ours. Also a scheme for refinement of parts of the phase space is applied. Nevertheless, the computation of the basin follows a different paradigm. Like in case of [NY97], also forward iterates of trajectories are analyzed. Indeed, the two approaches are similar if taken the *simple cell mapping* method of Hsu, which maps a cell only to one other, taking the center of the cell as scan point. Another method for cell mapping is *generalized cell mapping* [Hsu87, GK99], which deals with probability distribution among the image cells. Furthermore, there is *interpolated cell mapping* [TG88, HT04], a combination of simple and generalized cell mapping.

The simple and interpolated cell mapping approaches have in common that, in contrast to our approach, a lower bound can not be computed and the attractors can not be explicitly selected. In fact, attractors are detected by finding periodic cells. In case of an attractor which is not a periodic orbit, such a detection might not produce correct results. Parts of the attractor may be detected as belonging to the domain of attraction, or an attractor may not be detected as

a whole but as several coexisting cycles of periodic cells. These problems do not exist in our approach because we select attractors not by the localization of periodic cells but by the detection of components of the chain recurrent set.

The generalized cell mapping approach can be considered close to ours. Indeed, recurrent cells can be located as attractors, and the detection of the basin is also based on a breadth-first search. However, this approach requires additional amount of computation to construct the probability and also increases the computational complexity. Note also that the refinement of the phase space is based on a different approach. An area of investigation M is covered by a mesh of simplices which is locally refined by an iterative process. Hence, always the complete area M is subject of investigation, whereas in our approach only those parts of M are investigated which are covered by the current covering C^s and contain the solution of the investigation.

Next we look at the computation of attractors, repellers and filtrations. The simple approach of forward iterates is not capable of computing the complex attractors whose construction was proposed in Sec. 5.3, nor is it capable of computing filtrations or repellers. Only minimal attractors can be approximated, and even the localization of all coexisting minimal attractors can not be guaranteed. By a set-oriented method proposed by Dellnitz and Junge [DH97], the computation of relative global attractors can be achieved. As in our approach, also a discretization of the phase space is applied, and the images of sets are calculated. However, filtrations or attractors which are embedded into each other can not be computed. Order relations between components of the chain recurrent sets are also detected by [Eid95, Mis02] though in a different context, and not for the construction of attractors, repellers and filtrations. Note that in the same work also connecting orbits are mentioned but not explicitly computed.

Several methods for the computation of stable and unstable manifolds exist, see Sec. 2.5. Some of these methods produce better results than ours in terms of accuracy and performance. However, these methods can only localize stable manifolds in case the inverse is computable. Hence, even in case the mapping is invertible, the stable manifold might not be computed if the Newton method can not be applied. An exception is the approach by England *et. al.* [EKO05] but it is limited to the computation of 1-dimensional manifolds whereas our proposed approach can also be applied to compute higher dimensional manifolds of continuous mappings, no matter if they are noninvertible. We like to mention that the method by Dellnitz *et. al.* [DH97] uses a set-oriented approach for the computation of the manifolds close to ours. However, also for this approach, the computation of the system function's inverse is required in order to get the stable manifold.

In terms of connecting manifolds, there exists also a set-oriented approach by Dellnitz *et.al.* [DJT01]. The technique works similar to ours but, however, the computation of the inverse is required whereas our method can also be applied in case the inverse does not exist or is not computable.

5.7 Numerical Case Studies

In this section we present results acquired by the use of the algorithms described in this chapter. Besides verifying the correctness of the implementation it is also our intention to show the reader how to work with the methods in practice. He should get a deeper insight into the possibilities of our methods as well as their proper and effective usage. We will give examples for both types of dynamical systems, those discrete and those continuous in time.

5.7.1 Duffing System

We start with a well-known dynamical system continuous in time, the Duffing oscillator [AFH94, GH83, LR03]. We take a simple version described by the perturbed Duffing equation

$$\ddot{x} - x + x^3 + \varepsilon \dot{x} = 0,$$

see also Guckenheimer [GH83]. Its corresponding system is of the form

$$\begin{aligned} &\dot{\mathbf{x}}(t) = \mathbf{F}_{Duf}(\mathbf{x}(t)), \quad \mathbf{F}_{Duf} : \mathbb{R}^2 \to \mathbb{R}^2, \quad \mathbf{x} = (x,y)^T \\ &\mathbf{F}_{Duf}(\mathbf{x}) = \begin{pmatrix} y \\ x - x^3 - \varepsilon y \end{pmatrix} \end{aligned} \tag{5.36}$$

The parameter ε is set to $\varepsilon = 0.15$ in all our examples. The system has three equilibria, $E_1 = (-1,0)$, $E_2 = (1,0)$ and $E_3 = (0,0)$. The points E_1 and E_2 are attractors, the stable manifold of E_3 separates the domains of attraction, $W^s(E_1)$ and $W^s(E_2)$. We intend to construct one of these basins, $W^s(E_1)$, and some sequences of a filtration containing E_1, E_2 and E_3.

In order to detect the basin of E_1, we choose $M = [-2.0;2.0] \times [-2.0;2.0]$ as the area of investigation. This area M is initially divided into 20×20 boxes. In every subdivision a box will be divided into 4×4 new ones. We perform six subdivisions and compute the images $G^0, \ldots, G^6$. For a continuous system like the Duffing oscillator, we must additionally set parameters for the integration time Δt and the number of iterations n,see Sec. 4.1.1. In this case we set $\Delta t = 0.01$

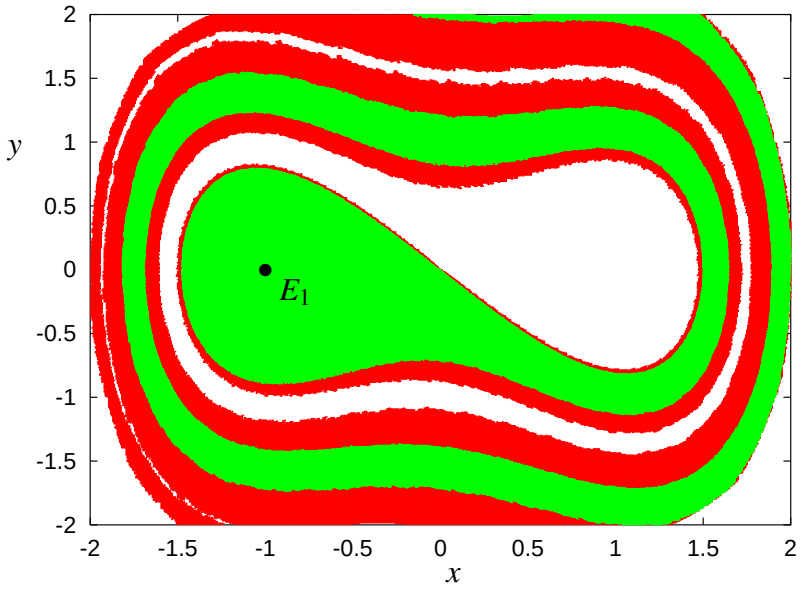


Figure 5.1: *Duffing system: The numerically calculated basin for E_1. We see the upper and lower bound, colored red and green.*

and $n = 20$. In order to get a basin, we must next define the selection of sets S_H which build the attractor L_S, see Eq. 5.13. We do this by giving a range $R = [-1.5;0.3] \times [-0.5;0.5]$ so that all recurrent cell sets $H_k \in \zeta$ which intersect with R get selected, see Eq. 5.15. We calculate the upper bound of the basin for this selection, $D_u(L_S)$, and mark all cells belonging to it for subdivision of the symbolic images $G^0, \ldots, G^4$. Note that R covers the equilibria E_1 as well as E_3. Reason is that in the first subdivisions only a rough approximation is required and our main concern is not to loose cells containing the border areas of the basin. Hence, we also select the cells around E_3. For the symbolic image G^5 we change the selection to $S_H(R')$ and use the range $R' = [-1.5;-0.1] \times [-0.5;0.5]$. The recurrent cell sets corresponding to E_3 are not included anymore. Furthermore, we set a flag so that only recurrent cell sets H_k with $Ex(H_k) = \emptyset$ are selected for S_H, see Eq. 5.16. Note that $L(S_H)$ is then an attractor of the symbolic image. We again calculate the upper bound for G^5. Only for the last symbolic image, G^6, the lower bound $D(L(S_H))$, is computed, see Eq. 5.19.

The results of the calculation can be seen in Fig. 5.1. The upper and the lower bound, $W_u(L(S_H))$ and $W(L(S_H))$, are shown. The area corresponding to the cells selected for S_H cover the immediate neighborhood of E_1. Note that due

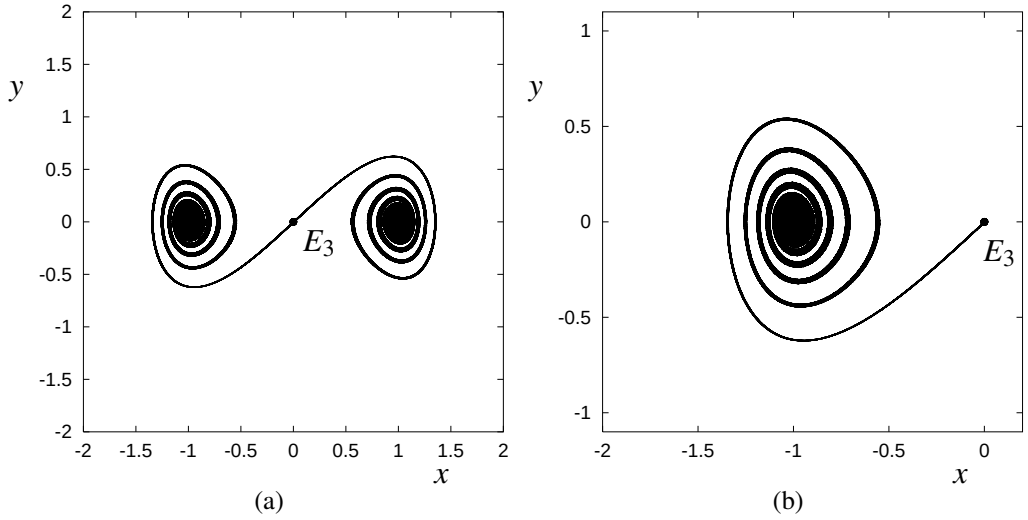


Figure 5.2: *Duffing system: (a) The numerically calculated boxes corresponding to the set $\mathcal{CL}(S_{E_3})$ for E_3, which forms an attractor. (b) The boxes for the connecting cells, $\mathcal{CS}(S_{E_3}, S_{E_1})$, between E_3 and E_1.*

to clustering, there are two boxes selected in the direct neighborhood of the fixed point E_1. The computation of these results takes around 8 minutes on our reference machine, and the symbolic images consist of up to 1 150 000 cells. Hereby, the construction of the graph takes the most time. The algorithms for the selection of S_H and the localization of the upper and lower bound of the basin require altogether only a few seconds of the total computation time.

Next we will construct a fine filtration on G for E_1, E_2, E_3 and show that its sequences are attractors. In the previous calculation we have already computed the attractor E_1. We can get E_2 on the same way. Furthermore, we can check that the point E_3 is also an equilibrium, but unstable. Hence, it can not be an attractor. Let us next consider clustering and define that a selection S_E is a set of recurrent cell sets H_k with recurrent cells $c_i \in H_k$ whose boxes $M(i)$ are scattered close around an equilibrium E. We can apply filtration and construct a B_2 so that

$$\phi = \{B_0 = \emptyset, B_1 = L(S_{E_1}) \cup L(S_{E_2}), B_2 = \mathcal{CL}(S_{E_3}), B_3 = V(G)\}$$

is a fine filtration and B_2 is an attractor. We show this in the following.

Ideally, to construct such a filtration we would first calculate and analyze the list of larger sets, $\mathcal{L}(H_k)$, in order to renumber the sets. Unfortunately, this can

not be done straightforwardly in a practical computation. Due to clustering, every equilibrium is not represented by a single cell in G but rather more by several cells, which are also separated in different recurrent cell sets H_k. Note that clustering usually can be observed to a larger extent in symbolic images built for systems continuous in time than in those built for systems discrete in time. In our case, the equilibria of E_1, E_2 and E_3 are scattered over around 70 recurrent cell sets. Hence, we will not analyze the larger cell sets but only build $\mathcal{CL}(S_{E_3})$. To do so, we construct the symbolic images $G_0,\ldots,G_8$ using the same parameter settings as before. This time we do not calculate the basin but $\mathcal{CL}(S_{E_3})$. Therefore, we take the selection $S_H(R)$ of recurrent cell sets intersecting with $R=[-0.1;0.1]\times[-0.1;0.1]$ to compute $\mathcal{CL}(S_H)$ for every G_s. In the first subdivisions, we also include selections around E_1 and E_2 to prevent the loss of required cells.

After the calculation has finished, we get $\mathcal{CL}(S_{E_3})$ as presented in Fig. 5.2(a). Hence, as expected, S_{E_3} is connected with S_{E_1} and S_{E_2}. So we can say that $S_{E_3}<S_{E_1}$, $S_{E_3}<S_{E_2}$. Furthermore, we can compute the result $O^+(\zeta)=\emptyset$ for G^8. Consequently, $\mathcal{CL}(S_{E_3})$ is an attractor and the filtration ϕ is correct. The processing takes around 12 minutes. Again, this is due to the construction process of the symbolic images which grow up to 1 200 000 cells.

If desired we can also produce the set $\mathcal{CS}(S_{E_3},S_{E_1})$ of all connecting cells between S_{E_3} and S_{E_1} as shown in Fig. 5.2(b). Therefore, we select the sets in the range $R_{E_3}=[-0.1;0.1]\times[-0.1;0.1]$ for E_3 and the ones in $R_{E_1}=[-1.1;0.9]\times[-0.1;0.1]$ for E_1. Then we detect the selection $\mathcal{CS}(S_H(R_{E_3}),S_H(R_{E_1}))$ for $G_0,\ldots,G_6$. The calculation takes around 7 minutes.

5.7.2 Ikeda Map

After the discussion of a continuous system follows now a system discrete in time, namely the Ikeda mapping [Ike79], which was already introduced in Sec. 3.6.1,see Eq. 3.51. All following numerical simulations have been carried out for the same parameter values $a=b=0.9$, $c_1=0.4$, $c_2=6.0$ and $r=0.9$. Recall that there exists a chaotic attractor $\mathcal{A}$, two unstable fixed points $P_{1,2}$ and a stable fixed point P_3. We calculate the basin for $\mathcal{A}$ and P_3 as well as some sequences of filtration.

In order to detect the domain of attraction for $\mathcal{A}$, the area $M=[-5.0;5.0]\times[-5.0;5.0]$ of the domain space was initially divided into 20×20 boxes. Later, every box is subdivided in 4×4 new ones. We construct $G^0,\ldots,G^3$. For $G^0,\ldots,G^2$ all recurrent cell sets in the range $R=[-2.0;2.0]\times[-3.0;2.0]$ are

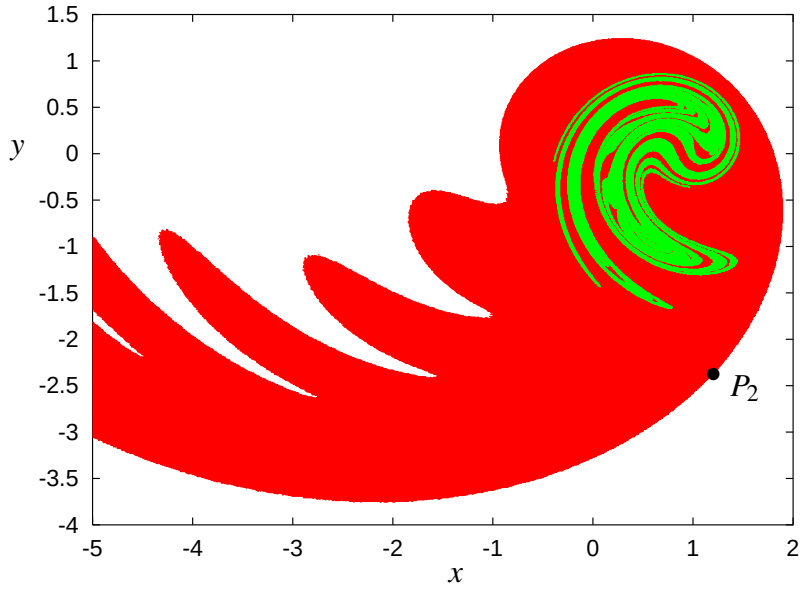


Figure 5.3: *Ikeda system: The numerically calculated basin (red) for the attractor $\mathcal{A}$ (green). For a better orientation, P_2 is also shown.*

selected as S_H and we subdivide all cells belonging to the upper bound $D_u(L_S)$. For the last graph G^3, we first analyze the recurrent cell sets and then explicitly choose the set H_0 as S_H. The boxes corresponding to H_0 are an outer covering of the attractor $\mathcal{A}$. The results of the calculation show that the set H_0 has no exit edges. Therefore it is an attractor on G^3. We finally compute the lower bound of its basin, $D(H_0)$.

The results of our computation can be seen in Fig. 5.3. The calculation takes less than 1 minute whereby the symbolic images consist of up to 350000 cells. The fast computation is achieved because we are dealing with a system discrete in time, and so the construction process of the symbolic image graph takes less time than for the Duffing system which is continuous in time.

Next the basin of the equilibrium P_3 within the domain $M = [-4.0;7.0] \times [-4.0;7.0]$ is computed. We select the recurrent cells sets with no exit edges in the range $R = [3.0;5.0] \times [2.0;4.0]$ as the attractor for the desired basin. In Fig. 5.4(a), the results of the computation are shown. We clearly see some numerical artifacts in the corners of the picture. These white areas should also belong to the basin. Such artifacts occur due to the limitations of symbolic

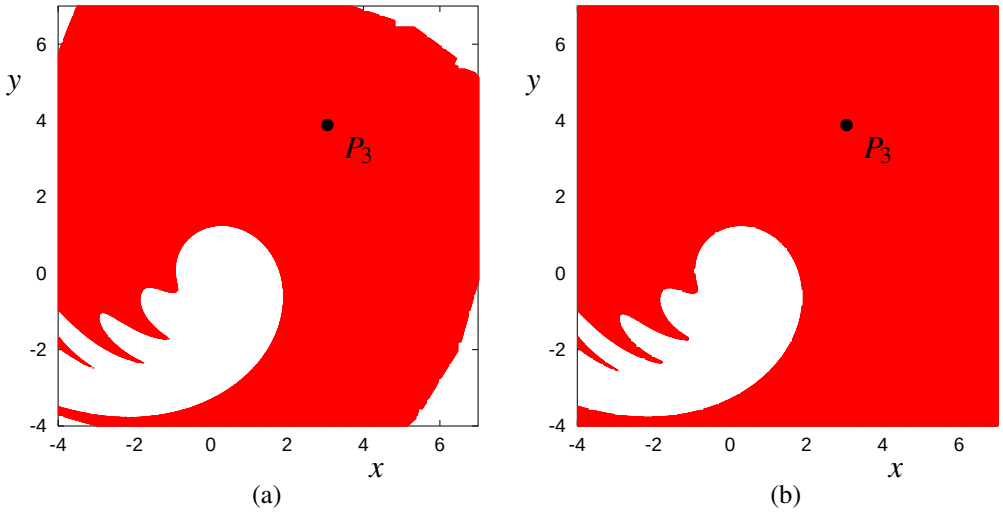


Figure 5.4: *Ikeda system: (a) The numerically approximated basin for P_3. (b) Computation by using the higher iterated function $\mathbf{f}_I^{[10]}$.*

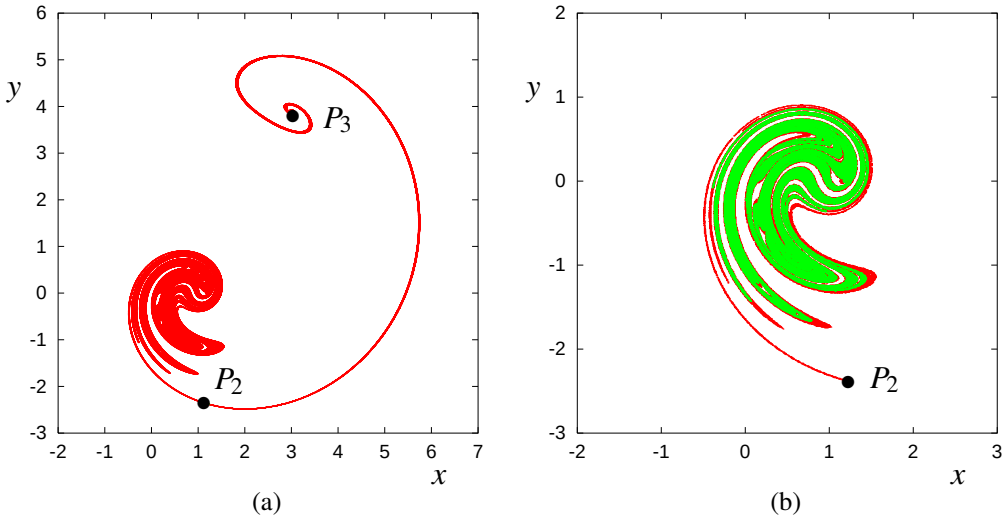


Figure 5.5: *Ikeda system: (a) The numerically calculated outer covering corresponding to the set $\mathcal{CL}(S_{P_2})$ (red) for P_2, which forms an attractor. (b) The outer covering for the connecting cells, $\mathcal{CS}(S_{P_2}, S_{\mathcal{A}})$ (red), between P_2 and $\mathcal{A}$ (green).*

image construction already mentioned in Remark 5.3. Trajectories started in these regions leave the area M covered by the symbolic image. Although they return to M and eventually finish in P_3, they are not computed as part of the basin because we can not follow their run after they have left M. In order to avoid such artifacts, some tuning techniques can be applied, see Sec. 4.2.1. More precisely, we use the higher iterated function $\mathbf{f}_I^{[10]}$ instead $\mathbf{f}_I$ as system function. By doing so, all images of scan points map into the area of investigation. The results of the computation are illustrated by Fig. 5.4(b). Note, however, that in this case the edges to the basin of $\mathcal{A}$ are not as smooth as for the computation without tuning.

H_k	$\lvert H_k\rvert$	$r(H_k)$			$\mathcal{L}(H_k)$
0	135 951	$[-0.372; 1.459]$	$\times$	$[-1.662; 8.625]$	-
1	1	$[1.188; 1.191]$	$\times$	$[-2.378; -2.375]$	0
2	1	$[1.191; 1.194]$	$\times$	$[-2.378; -2.375]$	0,1
3	1	$[1.194; 1.197]$	$\times$	$[-2.378; -2.375]$	1,2,0
4	1	$[3.000; 3.003]$	$\times$	$[3.894, 3.897]$	-
5	79	$[2.984; 3.022]$	$\times$	$[3.878; 3.909]$	4
6	1	$[1.200; 1.203]$	$\times$	$[-2.381; -2.378]$	4,5
7	1	$[1.197; 1.200]$	$\times$	$[-2.381; -2.378]$	4,5,6
8	1	$[1.206; 1.209]$	$\times$	$[-2.378; -2.375]$	4,5
9	1	$[1.203; 1.206]$	$\times$	$[-2.378; -2.375]$	4,5,8
10	1	$[1.200; 1.203]$	$\times$	$[-2.378; -2.375]$	4,5,8,9
11	1	$[1.197; 1.200]$	$\times$	$[-2.378; -2.375]$	$0,\ldots,5,8,10$
12	1	$[1.194; 1.197]$	$\times$	$[-2.381; -2.378]$	$0,\ldots,11$
13	1	$[1.194; 1.197]$	$\times$	$[-2.375; -2.372]$	0
14	1	$[1.200; 1.203]$	$\times$	$[-2.375; -2.372]$	$0,\ldots,5,8,\ldots,11,13$

Table 5.1: *Ikeda system: Numerically detected recurrent cell sets and their order relations.*

Next we consider the unstable point P_2. We expect that the relations $S_{P_2} < S_{P_3}$ and $S_{P_2} < S_{\mathcal{A}}$ exist, and approve that by calculating $\mathcal{CL}(S_{P_2})$ in 2 subdivisions. The initial covering consists hereby of 200×200 boxes, and in every subdivision step a box is divided in 4×4 new ones. The results are presented in Fig. 5.5(a). We analyze the larger sets $\mathcal{L}(S_{P_2})$ which we can acquire during the calculation of $\mathcal{CL}(S_{P_2})$. The computed data for G^3 is summarized in Tab. 5.1. As we can see, P_2 is, due to clustering, represented by more than one recurrent cell set so that $S_{P_2} = \{H_1, H_2, H_3, H_6 \ldots, H_{14}\}$. Furthermore, we have $S_{\mathcal{A}} = \{H_0\}$ and $S_{P_3} = \{H_4, H_5\}$. The investigation of the order relations leads to the conclusion that $S_{P_2} < S_{P_3}$ and $S_{P_2} < S_{\mathcal{A}}$, as expected. Additionally, the result $O^+(\zeta) = \emptyset$ can be

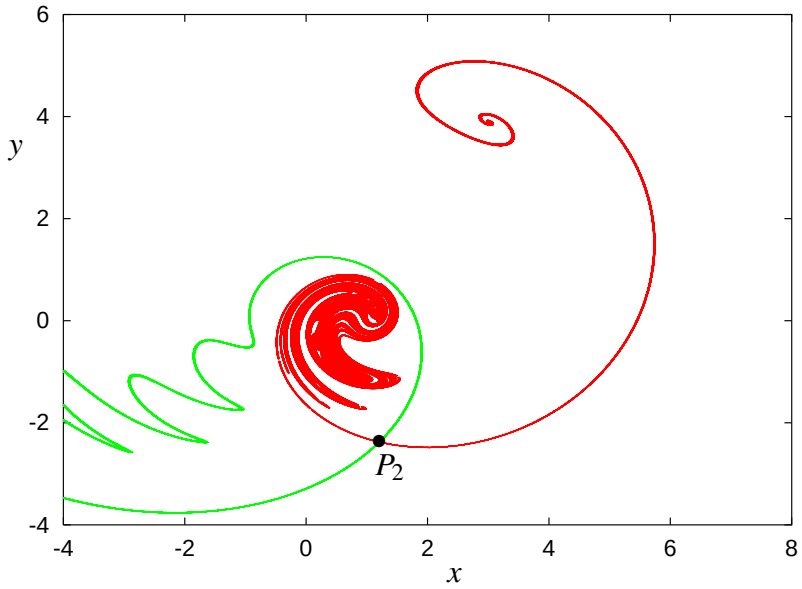


Figure 5.6: *Ikeda system: The numerically calculated outer covering of the stable (green) and unstable (red) manifolds of the saddle P_2.*

processed. Hence we have a filtration

$$\phi = \{B_0 = \emptyset, B_1 = L(S_{\mathcal{A}}) \cup L(S_{P_3}), B_2 = \mathcal{CL}(S_{P_2}), B_3 = V(G)\}$$

and B_2 is an attractor. Note that it is necessary to set an error tolerance, see Sec. 4.1.2, for this kind of computation. Otherwise, we would get $O^+(\zeta) \neq \emptyset$ because the symbolic image is approximated by only a few scan points. After a subdivision that leads to mappings from the outer covering of the attractor onto $M(\infty)$. This numerical artifact can be avoided if we set $e = 0.2$ for G^0, G^1 and $e = 0$ for G^2.

We also constructed the set $\mathcal{CS}(S_{P_2}, S_{\mathcal{A}})$ of all connecting cells between S_{P_2} and S_A as shown in Fig. 5.5(b). The area which corresponds to these connecting cells can be considered as an outer covering of the connecting manifold $C(P_2, \mathcal{A})$.

Furthermore, in Fig. 5.6 we present an outer covering of the stable and unstable manifolds for P_2. These manifolds can be acquired by computing the basin $D_u(L(S_{P_2}))$ of a G in order to approximate the stable manifold $W^s(P_2)$, and the inverse basin $D_u^{-1}(L(S_{P_2}))$ for the approximation of the unstable manifold $W^u(P_2)$. Note that for this kind of investigation only the upper bound can be computed. The

lower bound $D(L(S_{P_2})) = \emptyset$. Reason for this is that the (un)stable manifolds are 1-dimensional boundaries between the basins of attraction. Our coverings consist of 2-dimensional boxes. Hence, such boxes can not lie inside of the manifolds but only be a part of an outer covering.

Chapter 6

The RIM Method: Finding All Roots

In this chapter we introduce a new conceptual framework for the investigation of dynamical systems. The concept is called the *root-in-image* method, or simply the RIM method. Like symbolic analysis, this method is also based on multilevel phase space discretization. The main target is hereby not to construct a graph but to solve the root finding problem. Several kind of investigations can be reduced to such a root finding problem. See, for instance, the discussion in Sec. 2.2 concerning the localization of periodic points.

Root finding in general is a very well researched field if it comes to the localization of one solution of the system. However, in the context of our investigation tasks it is necessary that the method locates all existing roots. Successful approaches toward this direction were achieved in the field of *interval analysis*, see e.g. Alefeld and Herzberger [AH83], Hansen [Han92] or Kearfott [Kea96]. In some cases, also *homotopy methods* [CMPY78, AG90] can be applied, like in case the underlying function is polynomial [VH94]. There also exist approaches which are based on similar schemes like MPSD. See [YK89] in case the underlying function is holomorphic, and [HT90] in the context of global optimization. Furthermore, a set-oriented approach which combines a multilevel subdivision technique with a Newton-based iteration scheme was introduced by Dellnitz *et al.* [DSS02]. All these methods have in common that they are restricted to specific classes of functions, e.g. polynomials, and/or apply iteration schemes which are based on the multidimensional Newton method.

The advantage gained by the application of the Newton method is a fast convergence toward the solution. On the other hand, it requires that the Jacobian matrix is computed and that the system function $\mathbf{f}$ is smooth. In our approach, the usage of any Newton-based iteration scheme is avoided. Thus we do not have to calculate the Jacobian matrix nor any approximation of it. We are also not

limited to smooth functions. The only restriction is that $\mathbf{f}$ must be a C^0 function. Although the RIM method does usually not converge as fast to the detected solutions as some of the methods mentioned above, it finds roots with a high reliability, especially in the case that a large number of roots exist. We will also show that the performance of our approach is still competitive.

The development of the RIM method was motivated by the techniques of symbolic analysis. Although the construction and investigation of the symbolic image graph proved to be successful for many problems, there were also some shortcomings which are difficult to improve in the same conceptual framework. One of the main problems is clustering which can lead to inaccuracies concerning the precision of the results and to a high growth rate of cells during the subdivision process, see also Sec. 4.2. Another problem is the high computational complexity for the detection of periodic points, see Sec. 3.3.2 and Remark 3.6. The RIM method partly overcomes these shortcomings and generally provides better results than symbolic analysis for those kind of investigations it can be applied to.

In this chapter we focus on the basic task of the RIM method – the localization of roots. First the theory of the method is introduced, then some of the details regarding the implementation and the performance analysis are discussed. Afterwards, some numerical case studies are given and the method is compared with other root finding techniques. The application of the method in relation to the investigation of dynamical systems is then subject of the following chapter.

6.1 Theory of the Method

As already mentioned, the RIM method is based on the concepts of MPSD. Starting from this background, the theoretical framework for our method is constructed. We set the mathematical background, formulate the core algorithm and give proofs of correctness. The focus is put on how to identify those discretized areas which contain a solution and on the convergence of the method during the subdivision process. Whenever possible, we use a notation similar to that used in the preceding chapters.

6.1.1 The Core Algorithm

Let us consider a continuous mapping $\mathbf{f}$ on $\mathbb{R}^d$. Let the manifold M be a compact region in $\mathbb{R}^d$. The manifold M defines the region in which we want to detect roots of $\mathbf{f}$. More precisely, we say that we want to find the following set of points in M:

$$RO_f = \{\mathbf{x} \mid \mathbf{f}(\mathbf{x}) = 0,\ \mathbf{x} \in M \subseteq \mathbb{R}^d\}. \tag{6.1}$$

The main target of the RIM method is to describe an algorithm for the numerical approximation of the set RO_f by an outer covering. Therefore, we consider the discretization of the phase space. Let

$$C = \{M(1),...,M(n) \,|\, M(i) \subseteq M \text{ for } i = 1,\ldots,n\} \tag{6.2}$$

be a finite covering of compact and connected sets for the domain $M \subset \mathbb{R}^d$. Note that this definition slightly differs from those of Eq. 3.1. As for the symbolic image graph, the sets $M(i) \subseteq M$ are called boxes of the covering C, and i is the index of a box $M(i)$.

Let $\delta(M(i))$ be the diameter of a box according to Eq. 3.4. Every box $M(i)$ can be subdivided into a subset $\Upsilon_p(M(i))$ of p compact and connected sets $m(i,l), l = 1,\ldots,p$ so that

$$\Upsilon_p(M(i)) = \left\{ m(i,l) \,|\, \delta(m(i,l)) < \delta(M(i)), \bigcup_{l=1}^{p} m(i,l) = M(i) \right\}. \tag{6.3}$$

In order to find the roots in M, we use a multilevel subdivision scheme so that a covering C^{s+1} for each $s \in \mathbb{N}^+$ consists of the subdivided boxes of a set $R(C^s) \subseteq C^s$:

$$\Upsilon_p(C^s) = \bigcup_{M(i) \in R(C^s)} \Upsilon_p(M(i)), \tag{6.4}$$

$$C^{s+1} = \left\{ M(j) = m(i,l) \mid m(i,l) \in \Upsilon_p(C^s) \right\}, \tag{6.5}$$

whereby $C^0 = R(C^0) = \{M\}$, and $j = 1,\ldots,|C^{s+1}|$ is a new indexation for the subdivided boxes.

Such a subdivision scheme is in accordance with the concepts of *MPSD*. Hence, the main subject of the RIM method is the detection of the set $R(C^s) \subseteq C^s$. Due to our aims, a set $R(C^s)$ is a proper subset iff it contains all boxes $M(i) \in C^s$ in which at least one point $\mathbf{x} \in M(i)$ is a root for $\mathbf{f}$, i.e. $M(i) \cap RO_f \neq \emptyset$. In that case, the multilevel subdivision of M converges to a set of boxes which includes all roots of $\mathbf{f}$ in M. Each box $M(i) \in R(C^s)$ becomes for $s \to \infty$ an approximation of a root with a desired accuracy ε. We say that $R(C^s)$ is a *subdivision criteria*.

Definition 6.1. *We say that Υ_p is a* subdivision rule *on a covering C^0 if it is a relation according to Eq. 6.3 for compact and connected sets, and if an multilevel subdivision with Υ_p, according to Alg. 6.1, can be applied on C^0 to get C^s for every $s \in \mathbb{N}^+$.*

Algorithm 6.1 Finding all roots in M.

Require: M, $\mathbf{f}$, $n \in \mathbb{N}^+$
Ensure: $\bigcup R(C^n) \subset M$ contains all roots of $\mathbf{f}$

```
1:  R(C^0) ⇐ {M} {Initialize C^0}
2:  for all s = 1,...,n do {Get all boxes R(C^s) which contain roots}
3:     C^s ⇐ ∅
4:     for all M̃(j) ∈ R(C^{s-1}) do {Subdivide R(C^{s-1}) in C^s}
5:        C^s ⇐ C^s ∪ ϒ_p(M̃(j)) { Subdivide each box M̃(j) into new p boxes}
6:     end for
7:     R(C^s) ⇐ ∅
8:     for all M(i) ∈ C^s do {Detect R(C^s) ⊆ C^s}
9:        if hasRoot(f, M(i)) then {Detect if a box M(i) is subdivided}
10:          R(C^s) ⇐ R(C^s) ∪ {M(i)}
11:       else
12:          Delete M(i)
13:       end if
14:    end for
15: end for
```

We roughly outline the basic algorithm of our method as shown in Alg. 6.1. For an area M we construct an initial covering $C^0 = R(C^0) = \{M\}$, and subdivide it into some subsets $M(i) \in \Upsilon_p(M)$. Further details about the subdivision $\Upsilon_p(M)$ are discussed in Sec. 6.2. Note that $C^1 = \Upsilon_p(M)$. Then we select the subset $R(C^1) \subseteq C^1$ with all boxes $M(i) \in C^1$ according to the result of a function $\mathit{hasRoot}(\mathbf{f}, M(i))$ which decides if a box $M(i)$ will be subdivided or deleted. A subdivision of all cells $R(C^1)$ provides us the covering C^2. Again, we detect $R(C^2) \subset C^2$, and subdivide $R(C^2)$. We repeat this process until we reach a covering $R(C^n)$ which is considered to be a good approximation of the roots in M. The accuracy ε of our covering is decided by the largest diameter of the boxes $M(i) \in R(C^n)$, i.e.

$$\varepsilon = \max\left(\delta(M(i)) \mid M(i) \in R(C^n)\right).$$

In the following section we discuss the function $\mathit{hasRoot}(\mathbf{f}, M(i))$ in which the decision takes place if a box should be considered for further subdivision or not.

6.1.2 The Subdivision Criteria

The decision if a box $M(i)$ contains a root is the most integral part of our method. In Alg. 6.1 this is done by the function $\mathit{hasRoot}(\mathbf{f}, M(i))$. The basic idea of this function is to approximate the image of $M(i)$ with regard to $\mathbf{f}$ by a *convex hull*. Then we check if 0 lies inside this convex hull. If so, the function

hasRoot$(\mathbf{f}, M(i))$ returns `true` and the box $M(i)$ will be subdivided. If not, $M(i)$ is deleted. We start now first with some theoretical considerations.

For each box $M(i)$ we consider its image $IM(i)$ with respect to $\mathbf{f}(\mathbf{x})$ as

$$IM(i) = \mathbf{f}(M(i)) = \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in M(i)\}. \tag{6.6}$$

Due to the fact that $\mathbf{f}$ is continuous, and every $M(i)$ is a compact and connected subset of M, the image $IM(i)$ of a box $M(i)$ is also a compact and connected set.

Definition 6.2. *A box $M(i)$ is said to contain roots if $0 \in IM(i)$, and it does not contain any root if $0 \notin IM(i)$. We say that a box $M(i)$ might contain roots if $0 \in U$ for some neighborhood $U \supseteq IM(i)$.*

Obviously, if the image $IM(i)$ of the set $M(i)$ contains 0 than there exists at least one $\mathbf{x} \in M(i)$ which is a root of $\mathbf{f}$. Euclidean geometry is now applied to describe the property $0 \in IM(i)$. In the following we use the notation IM for an image $IM(i)$.

Due to the compactness of IM, we can introduce the following parameter. Let

$$\delta(IM) = \max(\rho(\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in IM) \tag{6.7}$$

be the diameter of an image IM whereby the function $\rho(\mathbf{x}, \mathbf{y})$ is the Euclidean distance. Additionally, we introduce

$$d(\mathbf{x}, P) = \min(\rho(\mathbf{x}, \mathbf{y}) \mid \mathbf{y} \in P) \tag{6.8}$$

as the distance between a point $\mathbf{x}$ and a set P.

Definition 6.3. **[Zie95]** *A set $Q \subset \mathbb{R}^d$ is called* convex*, if the line segment L between any two points $p, p' \in Q$,*

$$Q = \left\{\lambda p + (1-\lambda)p' \mid 0 \leq \lambda \leq 1\right\},$$

is also contained in Q. The convex hull $conv(Q)$ of a set $Q \in \mathbb{R}^d$ is the intersection of all convex sub-sets of $\mathbb{R}^d$ containing Q:

$$conv(Q) = \bigcap \left\{Q' \subset \mathbb{R}^d | Q \subset Q' \text{ and } Q' \text{ is convex}\right\}.$$

Definition 6.4. **[Zie95]** *Let $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$ be a finite non-empty set of points. Then the convex hull of P is given by*

$$conv(P) = \left\{ \sum_{i=1}^{n} \lambda_i p_i \mid \lambda_i \geq 0 \text{ for all } i \text{ and } \sum_{i=1}^{n} \lambda_i = 1 \right\}.$$

The convex hull for a finite non-empty set of points is the minimal convex set containing the given points. It can be imagined as a finite region of d-dimensional space enclosed by a finite number of hyperplanes, i.e. a polytope. Proofs and a more detailed discussion of convex hulls can be found in [Zie95].

Definition 6.5. *The set P_ε is called an* ε-reduced set *of IM if it is a finite non-empty subset of IM so that $\forall \mathbf{y} \in IM : d(\mathbf{y}, P_\varepsilon) \leq \varepsilon$. .*

For any $\varepsilon > 0$ such a P_ε of IM exists. Note that in a numerical computation a set P_ε is in general not known for an image $IM(i)$, nor can it be easily computed from its domain $M(i)$ by numerical methods. However, by taking a sufficiently large number of scan points $\mathbf{x} \in M(i)$ which are scattered all over the domain $M(i)$, we can get an approximation of a P_ε for a small ε. In Sec. 6.2 we give a more detailed analysis of this problem.

Consider now for a vector $\mathbf{x} = (x_1, \ldots, x_d)$ and an $\varepsilon > 0$ the element

$$B(\mathbf{x}, \varepsilon) = [x_1 - \varepsilon,\, x_1 + \varepsilon] \times \cdots \times [x_d - \varepsilon,\, x_d + \varepsilon]$$

as a box with an equidistant side length $l(B) = 2\varepsilon$. Then we can cover the image IM with a finite set of hypercubes:

$$A(P_\varepsilon) = \left\{ B_j = B(\mathbf{x}_j, \varepsilon) \mid \mathbf{x}_j \in P_\varepsilon \right\}.$$

Proposition 6.1. *If P_ε is an ε-reduced set of IM then $B_j \cap IM \neq \emptyset$ for every $B_j \in A(P_\varepsilon)$ and $\bigcup_{B_j \in A(P_\varepsilon)} B_j \supseteq IM$.*

Proof. The first property $B_j \cap IM \neq \emptyset$ is fulfilled due to the fact that for every $B_j = B(\mathbf{x}_j, \varepsilon)$ the point $\mathbf{x}_j \in B_j$ and $\mathbf{x}_j \in P_\varepsilon \subset IM$. The second property we prove by contradiction. Imagine that IM is not a subset of the union of boxes B_j. Then there exists a $\mathbf{x} \in IM$ so that $\mathbf{x} \notin B_j$ for every B_j. If $\mathbf{x} \in IM$ then there is a $\mathbf{x}_j \in P_\varepsilon$ with $\rho(\mathbf{x}, \mathbf{x}_j) \leq \varepsilon$. For each $\mathbf{x}_j$ there is a $B(\mathbf{x}_j, \varepsilon) \in A(P_\varepsilon)$ and, hence, $\mathbf{x} \in B(\mathbf{x}_j, \varepsilon)$. This is a contradiction. □

Next we define the set of all corners of a box B_j,

$$P(B_j) = \left\{ \mathbf{x} | \, \mathbf{x} \in B_j, \exists \mathbf{y} \in B_j : \rho(\mathbf{x}, \mathbf{y}) = \delta(B_j) \right\}.$$

Note that every $P(B_j)$ is a finite set with $|P(B_j)| = 2^d$. The corners of all boxes $A(P_\varepsilon)$ can then be described as

$$CP(P_\varepsilon) = \bigcup_{B_j \in A(P_\varepsilon)} P(B_j). \tag{6.9}$$

Proposition 6.2. *If P_ε is an ε-reduced set of IM then $\forall \mathbf{x} \in CP(P_\varepsilon) : d(\mathbf{x}, IM) \leq \varepsilon\sqrt{d}$.*

Proof. Obviously, every $\mathbf{x} \in CP(P_\varepsilon)$ belongs to a box $B_j = B(\mathbf{x}_j, \varepsilon)$. The point $\mathbf{x}_j \in B_j$ belongs to IM, and the most distant points from $\mathbf{x}_j$ in B_j are the points $\hat{\mathbf{x}} = \mathbf{x}_j \pm \varepsilon$. Hence, due to the definition of $B(\mathbf{x}_j, \varepsilon)$, the dimension d of the image space $IM \subset \mathbb{R}^d$ and the Euclidean distance ρ we get:

$$\begin{aligned} d(\mathbf{x}, IM) &\leq \rho(\mathbf{x}, \mathbf{x}_j) \leq \rho(\hat{\mathbf{x}}, \mathbf{x}_j) \\ &= \sqrt{\sum_{i=1}^{d} ((x_{j(i)} \pm \varepsilon) - x_{j(i)})^2} \\ &= \sqrt{\sum_{i=1}^{d} \varepsilon^2} \\ &= \varepsilon\sqrt{d}. \end{aligned} \tag{6.10}$$

□

Proposition 6.3. *If $A \subseteq B$ then $conv(A) \subseteq conv(B)$.*

Proof. We prove the proposition by contradiction. Assume there exist sets $A \subset B$ so that $\text{conv}(A) \nsubseteq \text{conv}(B)$. Then there is a point $\mathbf{x} \in \text{conv}(A) \setminus \text{conv}(B)$. Obviously, $\mathbf{x} \notin A$ because $A \subset B \subseteq \text{conv}(B)$. Due to the definition of a convex hull, such an $\mathbf{x}$ can be described by $\mathbf{x} = \sum_{i=1}^{n} \lambda_i p_i$, whereby $\sum_{i=1}^{n} \lambda_i = 1$ and $\{p_1, \ldots, p_n\} \subseteq A$. But then $\mathbf{x} \in \text{conv}(B)$ because $p_1, \ldots, p_n \in B$. This is a contradiction. □

Proposition 6.4. *If P_ε is an ε-reduced set of IM then $conv(IM) \subseteq conv(CP(P_\varepsilon))$.*

Proof. First we show that $B_j = \text{conv}(P(B_j))$ for every box B_j. Obviously, the set B_j is a box, and therefore a convex polytope so that $\text{conv}(B_j) = B_j$. Consider that $P(B_j)$ is a subset of B_j. Hence, $\text{conv}(P(B_j)) \subseteq B_j = \text{conv}(B_j)$.

The set $P(B_j)$ is defined as all pairs $\mathbf{x}, \mathbf{y} \in B_j$ which fulfill $\rho(\mathbf{x}, \mathbf{y}) = \delta(B_j)$. So each $\mathbf{x}_k \in P(B_j)$ can only be described by $\mathbf{x}_k = \lambda \mathbf{x}_k$ with $\lambda = 1$. Every $\mathbf{x}_l \in B_j \setminus P(B_j)$ fulfills for all $\mathbf{x}_k \in P(B_j)$ the property $\rho(\mathbf{x}_l, \mathbf{x}_k) < \delta(B_j)$ and hence $\mathbf{x}_l = \sum_{i=1}^{n} \lambda_k \mathbf{x}_k$ for some $\sum_{k=1}^{n} \lambda_k = 1$, $\lambda_k \geq 0$. In other words, $P(B_j)$ defines the set of vertices for the convex polytope B_j. It follows that $\text{conv}(P(B_j)) \supseteq B_j \Rightarrow \text{conv}(P(B_j)) = B_j$.

In the next step, we conclude that if $\text{conv}(P(B_j)) = \text{conv}(B_j) = B_j$ then also $\text{conv}(P(B_j) \cup P(B_k)) = \text{conv}(B_j \cup B_k)$ for any two boxes B_j and B_k. Recall that the convex hull of a set Q is the intersection of all convex sub-sets of $\mathbb{R}^d$ containing

Q, see Def. 6.3. Hence, for all convex sets Q' containing a $Q = P(B_j) \cup P(B_k)$ we can state:

$$\begin{aligned} P(B_j) \cup P(B_k) \subset Q' &\Leftrightarrow \text{conv}(P(B_j)) \cup \text{conv}(P(B_k)) \subset Q' \\ &\Leftrightarrow \text{conv}(B_j) \cup \text{conv}(B_k) \subset Q' \\ &\Leftrightarrow B_j \cup B_k \subset Q' \end{aligned} \tag{6.11}$$

The equality $\text{conv}(P(B_j) \cup P(B_k)) = \text{conv}(B_j \cup B_k)$ implies also that $\text{conv}(CP(P_\varepsilon)) = \text{conv}(\cup A(P_\varepsilon))$. Recall next that Proposition 6.1 proves that $IM \subseteq \cup A(P_\varepsilon)$. Finally, we can conclude from Prop. 6.3 that $\text{conv}(IM) \subseteq \text{conv}(\cup A(P_\varepsilon)) = \text{conv}(CP(P_\varepsilon)$. □

Theorem 6.1. *If for every $\varepsilon > 0$ the set P_ε is some ε-reduced set of IM then*

$$\lim_{\varepsilon \to 0} conv(CP(P_\varepsilon)) = conv(IM).$$

Proof. In Proposition 6.2 it was shown that $\forall \mathbf{x} \in CP(P_\varepsilon) : d(\mathbf{x}, IM) \leq \varepsilon\sqrt{d}$, hence we can say that for a distance

$$d(CP(P_\varepsilon), IM) = \max(d(\mathbf{x}, IM) \mid \mathbf{x} \in CP(P_\varepsilon))$$

the following inequality is fulfilled:

$$d(CP(P_\varepsilon), IM) \leq \varepsilon\sqrt{d}.$$

The value d is the dimension of $\mathbf{f}$ and, hence, a constant. So if $\varepsilon \to 0$ then obviously the distance between $CP(P_\varepsilon)$ and IM tends also to 0:

$$\lim_{\varepsilon \to 0} d(CP(P_\varepsilon), IM) = 0$$

Note now that for the distance between the convex hulls,

$$dCH(CP(P_\varepsilon), IM) = \max(d(\mathbf{x}, \text{conv}(IM)) \mid \mathbf{x} \in \text{conv}(CP(P_\varepsilon))),$$

the inequality $dCH(CP(P_\varepsilon), IM) \leq d(CP(P_\varepsilon), IM)$ is fulfilled. So it is easy to see that also

$$\lim_{\varepsilon \to 0} dCH(CP(P_\varepsilon), IM) = 0.$$

Note now that for any set P with $dCH(P, IM) = 0$ it follows that $\text{conv}(P) \subseteq \text{conv}(IM)$. Due to Proposition 6.4 we also know that $\text{conv}(CP(P_\varepsilon)) \supseteq \text{conv}(IM)$ is fulfilled. Hence we can state that:

$$\begin{aligned} &\lim_{\varepsilon \to 0} dCH(CP(P_\varepsilon), IM) = 0 \\ &\Rightarrow \left(\bigcap_{\varepsilon > 0} \text{conv}(CP(P_\varepsilon)) \subseteq \text{conv}(IM) \right) \wedge \left(\forall \varepsilon > 0 \ : \ \text{conv}(CP(P_\varepsilon)) \supseteq \text{conv}(IM) \right) \\ &\qquad \Rightarrow \lim_{\varepsilon \to 0} \text{conv}(CP(P_\varepsilon)) = \text{conv}(IM) \end{aligned}$$

□

Definition 6.6. *A convex hull* $conv(CP(P_\varepsilon))$ *is called a* covering hull *for IM if* P_ε *is an* ε*-reduced set of IM.*

Due to the theorems we stated above, it is obvious that a covering hull for every $IM(i)$ can be constructed from a set $P_\varepsilon \subset IM(i)$. Such a set P_ε can be approximated by the calculation of every $\mathbf{f}(\mathbf{x})$ for a finite set of points $\mathbf{x} \in M(i)$.

Theorem 6.2. *If* $conv(CP(P_\varepsilon))$ *is a covering hull for* $IM(i)$ *and* $0 \notin conv(CP(P_\varepsilon))$ *then the set* $M(i)$ *does not contain a root. If* $0 \in conv(CP(P_\varepsilon))$ *then* $M(i)$ *might contain a root.*

Proof. If $\text{conv}(CP(P_\varepsilon))$ is a covering hull for $IM(i)$ then $\text{conv}(CP(P_\varepsilon)) \supseteq \text{conv}(IM(i)) \supseteq IM(i)$. So if $0 \notin \text{conv}(CP(P_\varepsilon))$ then also $0 \notin IM(i)$ and, hence, $M(i)$ does not contain any root. On the other hand, if $0 \in \text{conv}(CP(P_\varepsilon))$ then $M(i)$ might contain a root because $\text{conv}(CP(P_\varepsilon)) \supseteq IM(i)$. □

Theorem 6.2 states that if we can construct a covering hull for $M(i)$ then we can find out if the domain $M(i)$ does not contain any root. However, due to the fact that the covering hull as well as $\text{conv}(IM(i))$ are usually proper supersets of $IM(i)$, in general we can not say that $M(i)$ does contain a root if $0 \in \text{conv}(CP(P_\varepsilon))$. We can only state that $M(i)$ might contain a root. But the better the approximation of $\text{conv}(CP(P_\varepsilon))$ the more domains $M(i)$ can be identified as containing no roots and deleted early in the subdivision process, which is essential for our algorithm. Furthermore, in the following chapter we will see that for higher subdivision steps the convex hull converges to the image $IM(i)$ and, hence, we eventually get only sets $M(i) \in R(C^s)$ which do contain roots.

6.1.3 Convergence of the Method

We show now that our method converges to the set of all periodic points. For a subdivision rule Υ_p on C^0 according to Def. 6.1 we define first the parameter

$$\delta C(s) = \max(\delta(M(i)) \,|\, M(i) \in C^s). \tag{6.12}$$

This parameter can be considered as the largest diameter of all boxes $M(i) \in C^s$. Note that $\delta C(s)$ describes the accuracy of the outer covering C^s. If $R(C^s)$ is a set of boxes so that each of them contains a root then these roots are approximated with an error of $e \leq \delta C(s)$.

For the diameter $\delta C(s)$ we can further say that

$$\lim_{s\to\infty} \delta C(s) = 0 \tag{6.13}$$

because for every subdivision s the diameter $\delta(m(i,l)) < \delta(M(i))$ for each new box $m(i,l) \in \Upsilon_p(M(i))$ of a $M(i) \in R(C^s)$, see Eq. 6.3 and, hence,

$$\delta C(0) > \delta C(1) > \delta C(2) > \ldots .$$

Obviously, if $\delta C(s)$ goes to 0 then the error $e \leq \delta C(s)$ of our approximation also goes to 0.

The next step is to prove that every box $M(i) \in R(C^s)$ does contain a root if $s \to \infty$. In Sec. 6.1.2 we showed that we can construct a covering hull $\text{conv}(CP(P_\varepsilon))$ around an image $IM(i)$. For $\varepsilon \to 0$ the covering hull converges to $\text{conv}(IM(i))$, see Theorem 6.1. Hence, it can be decided if $0 \in \text{conv}(IM(i))$. The subdivision criteria can therefore be defined as follows:

$$R(C^s) = \{M(i) \,|\, M(i) \in C^s \text{ and } 0 \in \text{conv}(IM(i))\}. \tag{6.14}$$

We prove now that a convex hull converges to $IM(i)$ for $s \to \infty$ so that we can eventually assure that every $M(i) \in R(C^s)$ does not only might contain a root but does contain a root. For this reason we introduce the parameter

$$\delta I(s) = \max(\delta(IM(i)) \,|\, M(i) \in C^s), \tag{6.15}$$

which describes the largest diameter of all images belonging to a covering C^s.

Proposition 6.5. *If Υ_p is a subdivision rule on C^0 then the following holds for every continuous function* $\mathbf{f}$*:*

$$\lim_{s\to\infty} \delta I(s) = 0$$

Proof. First we prove that $\delta I(s_1) \geq \delta I(s_2)$ for every $s_1 > s_2$. For every application of the subdivision rule Υ_p we subdivide a box $M(i) \in C^s$ in new boxes $m(i,l) \in \Upsilon_p(M(i))$. According to Eq. 6.3, $m(i,l) \subset M(i)$ and, hence, every $\mathbf{f}(m(i,l)) \subseteq \mathbf{f}(M(i))$, see also Eq. 6.6. Consequently, $\delta(\mathbf{f}(M(i))) \geq \delta(\mathbf{f}(m(i,l)))$ and so also $\delta I(s) \geq \delta I(s+1)$ for every s. We conclude $\delta I(s_1) \geq \delta I(s_2)$ for every $s_1 > s_2$.

Next we show that for every s_1 with $\delta I(s_1) > 0$ there exists a $s_2 > s_1$ so that $\delta I(s_1) \neq \delta I(s_2)$. We prove that by contradiction. Let us assume that for all $s > s_1$ the diameter $\delta I(s_1) = \delta I(s)$. If $\delta I(s_1) > 0$ then there must exist points $\mathbf{x}_1, \mathbf{x}_2 \in M(i) \subset C^{s_1}$ so that $\mathbf{x}_1 \neq \mathbf{x}_2$ and $\rho(\mathbf{f}(\mathbf{x}_1), \mathbf{f}(\mathbf{x}_2)) = \delta I(s_1) = \delta I(s)$ for all s. We focus now on two points $\mathbf{x}_1, \mathbf{x}_2$ which fulfill this property and have the smallest distance $\rho(\mathbf{x}_1, \mathbf{x}_2)$ of all points which fulfill the property. Without loss of generality, we assume there is only one such pair $\mathbf{x}_1, \mathbf{x}_2$. For $s \to \infty$ there exists now a s_2 so that $\delta C(s_2) < \rho(\mathbf{x}_1, \mathbf{x}_2)$ for these points $\mathbf{x}_1, \mathbf{x}_2$, see Eq. 6.13. Hence,

$\mathbf{x}_1 \in M(i) \in C^{s_2}$ and $\mathbf{x}_2 \in M(j) \in C^{s_2}$ with $M(i) \neq M(j)$. For every $\mathbf{x} \in M(i)$ with $\mathbf{x} \neq \mathbf{x}_1$ it counts that $\rho(\mathbf{f}(\mathbf{x}),\mathbf{f}(\mathbf{x}_1)) < \rho(\mathbf{f}(\mathbf{x}_1),\mathbf{f}(\mathbf{x}_2))$, and also for every $\mathbf{y} \in M(j)$ with $\mathbf{y} \neq \mathbf{x}_2$ is $\rho(\mathbf{f}(\mathbf{y}),\mathbf{f}(\mathbf{x}_2)) < \rho(\mathbf{f}(\mathbf{x}_1),\mathbf{f}(\mathbf{x}_2))$ because $\mathbf{x}_1,\mathbf{x}_2$ are the points with the smallest distance $\rho(\mathbf{x}_1,\mathbf{x}_2)$ which have a distance $\rho(\mathbf{f}(\mathbf{x}_1),\mathbf{f}(\mathbf{x}_2))$. Hence, $\delta I(s_2) < \rho(\mathbf{f}(\mathbf{x}_1),\mathbf{f}(\mathbf{x}_2)) = \delta I(s_1)$. This is a contradiction to our assumption, and so, obviously, there exists a $s_2 > s_1$ for every s_1 so that $\delta I(s_1) \neq \delta I(s_2)$.

Eventually, we can conclude that for every s_1 with $\delta I(s_1) > 0$ there exists a $s_2 > s_1$ so that $\delta I(s_1) > \delta I(s_2)$, and so

$$\lim_{s\to\infty} \delta I(s) = 0.$$

□

As the next step, we show that the distance between an image IM and its convex hull,

$$d(\text{conv}(IM), IM) = \max(d(\mathbf{x},IM) \,|\, \mathbf{x} \in \text{conv}(IM)),$$

tends to 0 if a subdivision scheme is applied and $s \to \infty$. This is stated by the next proposition. We first denote the set

$$\hat{C}^s = \{IM \mid IM = \mathbf{f}(M(i)) \text{ and } M(i) \in C^s\}.$$

Additionally, we introduce

$$dCHI(s) = \max(d(\text{conv}(IM),IM) \mid IM \in \hat{C}^s)$$

as the largest distance between an image $IM \in \hat{C}^k$ and its convex hull $\text{conv}(IM)$.

Proposition 6.6. *If Υ_p is a subdivision rule on C^0 then the following holds for every continuous function* **f***:*

$$\lim_{s\to\infty} dCHI(s) = 0$$

Proof. Due to the definition of a convex hull, $d(\text{conv}(IM),IM) \leq \delta(IM)$ and, consequently, $dCHI(s) \leq \delta I(s)$. In Proposition 6.5 we stated that $\lim_{s\to\infty} \delta I(s) = 0$, and so

$$\lim_{s\to\infty} dCHI(s) = \lim_{s\to\infty} \delta I(s) = 0.$$

□

Let $RO_f(M)$ be the set of all roots of the function **f** in the region M, and

$$U(C^s) = \bigcup_{M(i)\in R(C^s)} M(i)$$

the area selected by the subdivision criteria $R(C^s)$ as defined in Eq. 6.14. The following can then be stated.

Theorem 6.3. *If Υ_p is a subdivision rule on C^0 then for every continuous function* **f***:*

$$\lim_{s\to\infty} U(C^s) = RO_f(M).$$

Proof. For every s the set $R(C^s)$ contains all boxes $M(i) \in C^k$ which might contain a root, as described in Sec. 6.1.2. Hence, $RO_f(M) \subseteq U(C^s)$. Recall that we check in the subdivision criteria if $0 \in \text{conv}(IM)$ for every $IM \in \hat{C}^s$. So, applying the proposed subdivision criteria we could state that $M(i)$ does contain a root iff $\text{conv}(IM) = IM$. Recall now that Proposition 6.6 states $dCHI(s) \to 0$ as $s \to \infty$. It follows that then also $d(\text{conv}(IM), IM) \to 0$ for every $IM \in \hat{C}^s$ as $s \to \infty$. Hence, as $s \to \infty$ we can say that every $M(i) \in C^{s\to\infty}$ does contain a root. We remember that $\delta C(s) \to 0$ for $s \to \infty$, and so

$$\lim_{s\to\infty} U(C^s) = \bigcap_{s\geq 0} U(C^s) = RO_f(M).$$

□

Next we state that it is sufficient to compute for every IM an ε-reduced set instead of the convex hull $\text{conv}(IM)$ in order to converge to $P(M)$.

Definition 6.7. *We say that $P_{\varepsilon(e)}$ is an ε-reduced set of an image IM which depends on a fixed $e > 0$ so that $\varepsilon(e) \leq e \cdot \delta(IM)$.*

Theoretically, such an ε-reduced set can easily be constructed. Assume, for instance, that $e = 1$. Then for any $\mathbf{x} \in M(i) \in C^s$, the set $P_{\varepsilon(e)} = \{\mathbf{f}(\mathbf{x})\}$ with $\varepsilon(e) = \delta(IM)$ is an ε-reduced set of the image $IM = \mathbf{f}(M(i))$.

Let us now fix an $e > 0$. We define

$$dCHP(s,e) = \max(\, d(\text{conv}(\, CP(P_{\varepsilon(e)})\,),\, IM) \mid IM \in \hat{C}^s)$$

as the largest distance between an image IM belonging to a box $M(i) \in C^k$ and its covering hulls $\text{conv}(\, CP(P_{\varepsilon(e)})\,)$. The next proposition states that if a subdivision scheme is applied then the distance between an image $IM \in \hat{C}^s$ and any covering hull of it tends to 0 for $s \to \infty$.

Proposition 6.7. *Let us fix $e > 0$. If Υ_p is a subdivision rule on C^0 then the following holds for every continuous function* **f***:*

$$\lim_{s\to\infty} dCHP(s,e) = 0.$$

Proof. For every $P_{\varepsilon(e)}$ of every $IM \in C^s$ we can say that $\varepsilon(e) \leq e \cdot \delta(IM)$, and so also $dCHP(s,e) \leq e \cdot \delta I(s)$. In Proposition 6.5 we stated that $\lim_{s\to\infty} \delta I(s) = 0$, and so also $\varepsilon(e) \to 0$ as $s \to \infty$. According to Theorem 6.1, we know that $\lim_{\varepsilon\to 0} \text{conv}(CP(P_\varepsilon)) = \text{conv}(IM)$, and together with Proposition 6.6:

$$\lim_{s\to\infty} dCHP(s,e) = \lim_{s\to\infty} dCHI(s) = \lim_{s\to\infty} \delta I(s) = 0.$$

□

We denote now a modified subdivision criteria as follows:

$$R(C^s,e) = \left\{ M(i) \mid M(i) \in C^s \text{ and } 0 \in \text{conv}(\, CP(P_{\varepsilon(e)}(i))\,) \right\}, \tag{6.16}$$

where $P_{\varepsilon(e)}(i)$ is an ε-reduced set of an $IM(i) = \mathbf{f}(M(i))$. We assume that in the context of the subdivision criteria $R(C^s,e)$ only one such set $P_{\varepsilon(e)}(i)$ exists, i.e. is computed, for every $M(i)$. This $P_{\varepsilon(e)}(i)$ depends on $e, \mathbf{f}$ and C^s.

The area selected by the subdivision criteria $R(C^s,e)$ is denoted by

$$U(C^s,e) = \bigcup_{M(i)\in R(C^s,e)} M(i).$$

The following can then be stated.

Theorem 6.4. *If Υ_p is a subdivision rule on C^0 then for every continuous function* $\mathbf{f}$ *and a fixed $e > 0$:*

$$\lim_{s\to\infty} U(C^s,e) = RO_f(M).$$

Proof. Obviously, $U(C^s,e) \supseteq U(C^s)$ because $\text{conv}(\, CP(P_{\varepsilon(e)})\,) \supseteq \text{conv}(IM)$, see Proposition 6.4. Hence, also $RO_f(M) \subseteq U(C^s) \subseteq U(C^s,e)$. Recall that we check in the subdivision criteria if $0 \in \text{conv}(\, CP(P_{\varepsilon(e)}(i))\,)$ for every $M(i) \in C^s$. So, applying the proposed subdivision criteria we could state that $M(i)$ does contain a root iff $\text{conv}(\, CP(P_{\varepsilon(e)})\,) = IM$. Recall now that Proposition 6.7 states $dCHP(s,e) \to dCHI(s)$ and $dCHI(s) \to 0$ as $s \to \infty$. We remember that $\delta C(s) \to 0$ for $s \to \infty$, and so

$$\lim_{s\to\infty} U(C^s,e) = \bigcap_{s\geq 0} U(C^s,e) = \bigcap_{s\geq 0} U(C^s) = RO_f(M).$$

□

We see that the set $R(C^s,e)$ converges to the set $RO_f(M)$ for an infinite number of subdivisions s by applying the subdivision algorithm of Sec. 6.1.1 and the subdivision criteria described in Sec. 6.1.2.

Algorithm 6.2 Subdivision criteria hasRoot($\mathbf{f}, M(i)$).

Require: $\mathbf{f}$, $M(i)$, $\tilde{\varepsilon}$, e.
Ensure: Returns `true` if $M(i)$ might contain a root.

```
1: Calculate S_ε̃(M(i)) ⊂ M(i) {Calculate the scan points}
2: P̃(i) ⇐ ∅
3: for all x ∈ S_ε̃(M(i)) do {Get P̃(i) for the scan points S_ε̃(M(i))}
4:    P̃(i) ⇐ f(x) ∪ P̃(i)
5: end for
6: ε ⇐ e · δ(P̃(i)) { Approximate ε for P̃(i)}
7: Calculate CP(P̃(i)) for ε
8: Calculate conv(CP(P̃(i))) {Calculate the convex hull using Quickhull}
9: if conv(CP(P̃(i))) = conv(CP(P̃(i)) ∪ 0) then {See Eq. 6.17}
10:    Return true
11: else
12:    Return false
13: end if
```

6.2 Implementation Details

In this section we discuss the implementation of the RIM method. As for the construction of the symbolic image graph, the area M of investigation is a d-dimensional rectangle area and all coverings C^s consist of d-dimensional uniform grid boxes $M(I)$ whose positions are described by unique multi-indices $I \in \mathbb{N}^d$, compare with Sec. 3.2.1. A subdivision rule $\Upsilon_p(M(I^s))$ divides a box $M(I^s)$ into p new uniform grid boxes $\tilde{M}(I^{s+1})$ which have a new indexation $I^{s+1} \in \mathbb{N}^d$, see also Sec 3.2.3. The size and number of these boxes is user-defined and variable. For any selection of boxes $R(C^s)$ with $|C^s| = n_s$ such a subdivision can be done in linear time $O(n_s)$, see Remark 3.7.

Our main concern focuses now on the subdivision criteria as described in Sec. 6.1.2. In Alg. 6.1 this criteria is identified by the function *hasRoot*($\mathbf{f}, M(i)$). For its implementation, see Alg. 6.2, mainly two tasks require a further discussion. First, the approximation of an ε-reduced set P_ε for each $IM(i)$ as defined in Def. 6.5, and second, the construction of a covering hull for $IM(i)$ according to Def. 6.6.

In order to get a set P_ε we select a finite set of points belonging to $M(i)$. These points are denoted the scan points $S_{\tilde{\varepsilon}}(M(i)) \subset M(i)$, see also Eq. 3.27. Then the set P_ε of $IM(i)$ is constructed by calculating $\mathbf{f}(\mathbf{x})$ for each $\mathbf{x} \in S_{\tilde{\varepsilon}}(M(i))$. The set

$$\tilde{P}(i) = \{\mathbf{f}(\mathbf{x}) \,|\, \mathbf{x} \in S_{\tilde{\varepsilon}}(M(i))\},$$

is then an ε-reduced set, though we do not know the size of its ε. Hence, we consider $\tilde{P}(i)$ only as an approximation of an ε-reduced set, and say that $\tilde{P}(i)$ is a precise approximation if ε is considered to be small. Empirically, we know that an approximation $\tilde{P}(i)$ is usually sufficiently precise for our purposes if the scan points $S_{\tilde{\varepsilon}}(M(i))$ are chosen properly. A proper selection of scan points can be achieved by taking mainly points of the box's boundary, and, additionally, some of its interior. These points should lie on a fixed grid so that

$$\forall \mathbf{y} \in M(i) : d(\mathbf{y}, S_{\tilde{\varepsilon}}(M(i))) \leq \tilde{\varepsilon}$$

for a user-defined $\tilde{\varepsilon}$. Note, that the smaller $\tilde{\varepsilon}$ is chosen the better is the approximation $\tilde{P}(i)$. The selection of a proper $\tilde{\varepsilon}$ depends heavily on the properties of the system function **f** and is therefore the task of the user. In Sec. 7.1.2 we present some standard settings and guidelines for the selection of scan points which proved to be successful for a large number of systems. As already mentioned, a different method for the selection of scan points in a set, which is based on the estimation of Lipschitz constants, was given by Junge, see [Jun00]. We believe that a similar approach might also lead to a rigorous detection of an ε-reduced set, though this is not yet implemented in our software.

The second task to focus on is the construction of the covering hull of $IM(i)$, depending on the ε-reduced set $\tilde{P}(i)$, see Def. 6.6. As a first step, this implies the computation of the set $CP(\tilde{P}(i))$, see Eq. 6.9. In order to do so, the value ε of the ε-reduced set $\tilde{P}(i)$ would be required. But, as stated above, we usually do not know ε for $\tilde{P}(i)$. Hence, ε must be approximated. In our implementation we offer two approaches to do so. First, the user sets a general value e, and each $\varepsilon(e)$ is then approximated by $\varepsilon(e) = e \cdot \delta(\tilde{P}(i))$. For a large e this strategy assures that we do not loose any box containing roots. As a second approach, we simply set $\varepsilon = 0$ because, in practice, $\text{conv}(\tilde{P}(i))$ proved to be a sufficiently good approximation of $\text{conv}(IM(i))$ if $\tilde{P}(i)$ is a sufficiently precise approximation of $IM(i)$. In order to clarify this, we recall that $\tilde{P}(i)$ is a precise approximation of $IM(i)$ if ε is small. This implies that the difference between $CP(\tilde{P}(i))$ and $\tilde{P}(i)$ is also small, see Proposition 6.2, and we can conclude that $CP(\tilde{P}(i)) \approx \tilde{P}(i)$. In Sec. 7.1.2 we will compare both approaches in more detail and give examples.

If ε is approximated, the calculation of $CP(\tilde{P}(i))$ can easily be achieved, and we can next compute the convex hull of $CP(\tilde{P}(i))$. To calculate a convex hull we use a standard software package called Quickhull, see [BDH96]. For an input set of finite points $CP(\tilde{P}(i))$, Quickhull computes the convex hull $\text{conv}(CP(\tilde{P}(i)))$. The Quickhull software additionally allows us to check our subdivision criteria

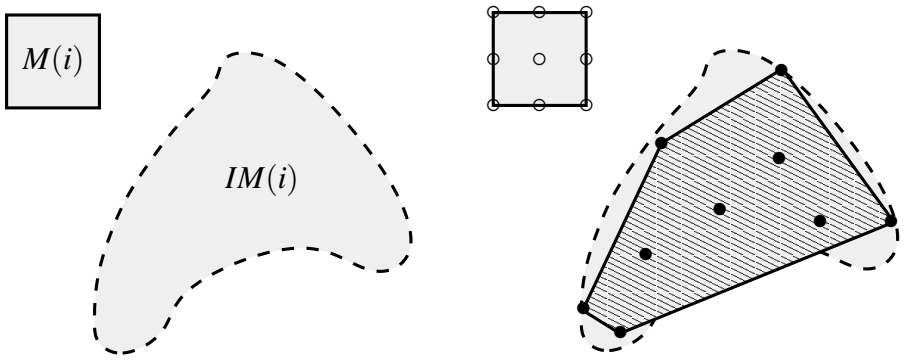


Figure 6.1: *Approximation of the image $IM(i)$ of a box $M(i)$. (a) The box $M(i)$ and its image $IM(i)$. (b) The scan points $\mathbf{x}_j \in S_{\tilde{\varepsilon}}(M(i)) \subset M(i)$ are marked with $\circ$. The images of these points $\mathbf{y}_j \in CP(\tilde{P}(i)) \subset IM(i)$ are marked with $\bullet$. The convex hull $\text{conv}(CP(\tilde{P}(i)))$ is shown as a hatched area.*

$0 \in \text{conv}(CP(\tilde{P}(i)))$, see Theorem 6.2, by testing if

$$\text{conv}(CP(\tilde{P}(i))) = \text{conv}(CP(\tilde{P}(i)) \cup 0). \tag{6.17}$$

If this condition is fulfilled, the box $M(i)$ belongs to $R(C^k)$, otherwise not.

Example 6.1. *Fig. 6.1 illustrates the approximation of the image $IM(i)$ of a box $M(i)$. A set of scan points $\mathbf{x}_j \in S_{\tilde{\varepsilon}}(M(i)) \subset M(i)$ is selected, marked with $\circ$ ($j = 1,\ldots,9$). These points are mapped onto the points $\mathbf{y}_j = \mathbf{f}(\mathbf{x}_j)$, marked with $\bullet$. We assume $e = 0$. Then the set $CP(\tilde{P}(i))$ is given by*

$$CP(\tilde{P}(i)) = \tilde{P}(i) = \{\mathbf{y}_1,\ldots,\mathbf{y}_9\},$$

and the convex hull $\text{conv}(CP(\tilde{P}(i)))$ can be constructed. In Fig. A.3(b), this convex hull $\text{conv}(CP(\tilde{P}(i)))$ is shown as a hatched area. As one can see, two problems can arise. Firstly, $\text{conv}(CP(\tilde{P}(i)))$ does not cover the complete area $IM(i)$. If $0 \in IM(i) \setminus \text{conv}(CP(\tilde{P}(i)))$ then the box $M(i)$ is deleted although it contains a root. Secondly, $\text{conv}(CP(\tilde{P}(i)))$ covers an area outside of $IM(i)$. If $0 \in \text{conv}(CP(\tilde{P}(i))) \setminus IM(i)$ then the box $M(i)$ is subdivided although it does not contain a root.

Although the computation of convex hulls in Quickhull is not limited by the dimension d of the input point set $CP(\tilde{P}(i))$, it must be considered that the performance heavily depends on d. Let $n = |CP(\tilde{P}(i))|$ be the number of input points, r

be the number of processed points and f_r be the maximum number of facets of r vertices ($f_r = O(r^{\lfloor d/2 \rfloor}/\lfloor d/2 \rfloor !)$ [Kle66].

Definition 6.8. [BDH96] *An execution of Quickhull is* balanced *if the average number of new facets for the j-th processed point is df_j/j and if the average number of partitioned points for the j-th processed point is $d(n-j)/j$.*

We assume that Quickhull is usually balanced in our application and consider the following theorem.

Theorem 6.5. [BDH96] *Let n be the number of input points in R^d, and r be the number of processed points. If the balance conditions hold, the worst case complexity of Quickhull is $O(n\, log(r))$ for $d \leq 3$ and $O(nf_r/r)$ for $d \geq 4$.*

As the dimension d increases, the number of facets in a convex hull grows rapidly for the same number of vertices. For example, the convex hull of 300 cospherical points in 6-d has $30,000$ facets [BDH96]. Although complexity is almost linear for low d, it increases dramatically for higher dimensions. Note now that our method requires the calculation of a convex hull for every $M(i) \in C^s$ and, hence, is only applicable for lower dimensions. Practical experience shows that our approach can be used for dimensions $d \leq 6$.

6.3 Performance Analysis

We discuss now the performance of the RIM method in general. In our implementation, the number of scan points in a box $M(i)$ is fixed to a user-defined constant l so that for every set of scan points $|S_{\tilde{\varepsilon}}(M(i))| = l$. Hence, the performance of the subdivision criteria depends only on the number l.

Proposition 6.8. *The application of the subdivision criteria on a box $M(i) \subset \mathbb{R}^d$, see Alg. 6.2, is in $O(l\, log(r))$ for $l \leq 3$ and $O(l\ f_r/r)$ for $d \geq 4$ if the balance condition holds for every convex hull calculation.*

Proof. Obviously, the calculation of $\tilde{P}(i)$ depends only on the number of scan points l. The computation of $CP(\tilde{P}(i))$ is a linear operation as well and, hence, also in $O(l)$. So the main computational effort arises from the construction of the convex hull. Recall that the number of input points $CP(\tilde{P}(i))$ for the convex hull computation depends on the number of scan points c and is in $O(l)$. Consequently, the complexity of the subdivision criteria is governed by the complexity of the convex hull computation, and thus can be described as in Theorem 6.5 with input size l. □

Theorem 6.6. *The calculation of a set $R(C^{s+1})$ for an input $R(C^s)$ with $|C^s| = n_s$ according to Alg. 6.1 is in $O(n_s)$.*

Proof. According to Remark 3.7, the subdivision of $R(C^s)$ in C^{s+1} with $|R(C^s)| = n'_s \leq n_s$ is in $O(n_s)$. In order to get $R(C^{s+1})$ from C^{s+1} , the subdivision criteria must be applied on every box $M(i) \in C^{s+1}$ with $|C^{s+1}| = p \cdot n'_s$, whereby p depends on the subdivision rule Υ_p. We stated that the subdivision criteria only depends on the number of scan points in a box, see Proposition 6.8. Hereby, the number of scan points is a fixed constant l. Hence, the computation time of a subdivision criteria is also fixed for every box $M(i)$, and therefore in $O(l\ f_r/r) \subseteq O(l\ f_p/p) \subseteq O(1)$. The calculation of $R(C^{s+1})$ requires the computation of the subdivision criteria for every $M(i) \in C^{s+1}$. Consequently, the complexity is $O(l \cdot p \cdot n'_s) \subseteq O(n_s)$. □

It can be concluded from the above theorem that our algorithms behave quite well in terms of complexity. Depending on the number of boxes n_s, the whole subdivision process to get $R(C^{s+1})$ requires only linear computation effort. However, the number of boxes n_s can grow exponentially during the subdivision process. We show this in the following.

Definition 6.9. *We say that our method converges under optimal conditions if every $M(i) \in R(C^s)$ for every s does contain roots.*

If the optimal conditions are fulfilled then only boxes get subdivided which are of relevance and every n_s is minimal. Consider now that we converge under optimal conditions. Then the number n_s only depends on the number r of roots in M because we subdivide not more than one box per root in every subdivision s, and hence, get $n_s \leq r$. We first consider a best case scenario – there is only one root in M. Then we would have $n_s = 1$, and converge in $O(1)$. However, in a worst case scenario the number of boxes n_s can grow exponentially during the subdivision process – even under optimal conditions. We show this by an example. Let $\mathbf{f}$: $\mathbb{R}^d \to 0$ be a continuous function which maps every point to 0. Hence, every $\mathbf{x} \in M \subset \mathbb{R}^d$ is a root, and no box will be deleted during the subdivision process. Let further Υ_p be a subdivision rule which subdivides each box $M(i)$ into 2 new boxes. Then

$$n_s = |R(C^s)| = 2^s.$$

Obviously, in that case we have an exponential growth rate for n_s.

Consider next that convergence under optimal conditions (or almost optimal conditions) can in general not be guaranteed. The property depends heavily on the characteristics of the investigated system function $\mathbf{f}$. In case that the optimal conditions are not fulfilled, we will still converge to a $R(C^s)$ which only has boxes

with roots, as shown in Sec. 6.1.3 – but n_s is not minimal and the complexity of the computation increases. This might especially happen for low s and, in worst case, can also lead to an exponential increase of n_s during the first subdivision steps.

So we see that the performance of our method mainly depends on the characteristics of the system function. In best case we have a complexity of $O(1)$, and in worst case $O(2^n)$.

Next we look on the convergence rate for the root approximation. As mentioned before, the error e is defined by the diameter of the boxes $M(i)$. Let Υ_p be a subdivision rule which divides every box $M(i)$ in $p = 2^d$ new boxes with a diameter $\frac{1}{2} \cdot \delta(M(i))$. Then the diameter shrinks exponentially:

$$e_s \leq \delta C(s) = \frac{1}{2^s} \cdot \delta(M). \tag{6.18}$$

In terms of root finding, we would say that our method converges *linearly* to the roots. Notice that this only counts under optimal conditions. In contrast, we know that the Newton-based methods converge *superlinearly*, and so, as mentioned before, the convergence rate of our method toward a solution can not compete with those of the Newton-based methods.

We also like to mention that the RIM method can be combined with a Newton-like iteration scheme in order to improve performance and achieve a higher accuracy of the solutions. This involves the application of two computational steps. Firstly, the RIM method is applied to compute an outer approximation $R(C^s)$ of the roots for a $s \geq 0$. Then, secondly, for each $M(i) \in R(C^s)$ an $\mathbf{x} \in M(i)$ is chosen as the initial value for the application of a Newton-like iteration scheme. If $M(i)$ is a reasonably small outer covering of a root then $\mathbf{x} \in M(i)$ is in its linear neighborhood, and the Newton method converges toward the root. Hereby, each $M(i) \in C^s$ should be an outer covering of exactly one root.

6.4 Comparisons and Numerical Case Studies

As stated above, there exist several other global root finding techniques. We compare our approach with some of these methods by the means of numerical case studies. The focus is on those techniques which can be applied to a broad class of functions, namely interval analysis as implemented by Kearfott [Kea96], the set-oriented approach of Dellnitz *et al.* [DSS02], and the selection of random

points.

Global root finding by interval analysis is based on interval arithmetics, see e.g. [AH83]. Hereby, an area $M(i)$ in the state space M, which is in our context a box, is considered to be an interval vector. On these intervals, an *interval Newton method* is applied in order to find roots. The main target hereby is to let those interval vectors which contain a solution shrink, so that they provide a reasonably small outer covering of the roots. An advantage of this approach is that a rigorous computation of an outer covering which contains all roots is generally possible. However, the practical application is limited by the fact that interval computations can produce large error bounds. This has a serious impact on the distinction if there is a root in a box $M(i)$, and if this is the only root in the box, see e.g. [Kea97]. As a result, there might be clusters of boxes around a solution and/or large interval vectors $M(i)$ which do not provide a good approximation of the root(s) they contain. Note, furthermore, that interval analysis depends on the application of a Newton-like method and requires also the computation of the Jacobian matrix or an approximation of it.

The set-oriented approach by Dellnitz *et al.* [DSS02] is based on the same concepts as the RIM method – discretization of the phase space by boxes and multilevel subdivision of these boxes. Nevertheless, there are fundamental differences. The underlying idea of the approach by Dellnitz *et al.* [DSS02] is to view a Newton-based iteration scheme as a dynamical system, and then localize the fixed points of this system. More precisely, assume N_f is some Newton-based iteration scheme for the underlying system $\mathbf{f}$. Then, instead of finding a root by iteration of N_f for an initial guess, N_f is seen as a dynamical system, and the fixed points of this system are all the roots of $\mathbf{f}$. The detection of the fixed points of N_f is achieved by application of set-oriented methods for the localization of attractors and invariant sets [DH97, DJ98]. These methods were already mentioned earlier in the context of symbolic analysis. Indeed, considering N_f as the system function, then its fixed points could also be localized by investigation of the symbolic image graph. The RIM method significantly differs from this approach in the way that a Newton-based iteration scheme is not considered at all. Instead, roots are localized by the computation of the convex hull of the image of boxes $M(i)$.

Let us now focus on the practical application of the RIM method. As a first approach, we tried to solve the systems presented by Kearfott [Kea87]. These systems are used as examples for the `INTBIS` [KN90] and `GLOBSOL` [kea] software. Except for the high-dimensional systems with $d > 6$, we were able to find the roots for all these nonlinear equations. The RIM method

even converged under optimal conditions. However, for such a scenario our approach, though competitive, is not advantageous because only a small number of roots existed. Therefore, we give a detailed analysis for another test system.

In Dellnitz *et al.* [DSS02] a global root finding approach was introduced, and some examples given for its usage. We discuss one of these examples and compare the results. The system is defined as follows:

$$\mathbf{f}_D : \mathbb{R}^2 \to \mathbb{R}^2, \quad \mathbf{x} = (x,y) \tag{6.19}$$

$$\mathbf{f}_D(\mathbf{x}) = \left(\varphi(y) \prod_{i=1}^{40} (x - x_i),\ \phi(x) \prod_{i=1}^{40} (y - y_i) \right)$$

$$\text{with} \quad x_i = y_i = \begin{cases} 0.1 + \frac{-10+i}{1000}, & \text{for} \quad i = 1 \ldots 20, \\ 0.9 + \frac{-30+i}{1000}, & \text{for} \quad i = 21 \ldots 40 \end{cases}$$

$$\text{and} \quad \varphi(y) = \sin(4y),\ \phi(x) = \sin(4x).$$

According to Dellnitz *et al.* [DSS02], the system has 1649 roots in the domain $M = [-3.0;3.0] \times [-3.0;3.0]$. Most of these roots are concentrated in four small clusters. In the original work, the method of Dellnitz was compared with the global root finding routine `c05pbc()` of the `NAG` library [nag]. For 10000 initial random points, the `c05pbc()` routine found only 300 roots while the set-oriented method of Dellnitz found a covering of all roots. However, in the approach of Dellnitz, after 24 subdivisions only a rough approximation of the four clusters was found. So it was still necessary to switch to a Newton-based method to get the exact roots.

If we apply now the RIM method to Eq. 6.19 with an initial covering $M = [-3.0;3.1] \times [-3.0;3.1]$, a subdivision of each box into 2×2 new ones, and $l = 11 \cdot 2^2 + 1$ scan points per box, we converge straight to all 1649 roots. We even converge under optimal conditions, and have never more than 1649 boxes per subdivision step. In Fig. 6.2 we present the results of our computation. We calculated the boxes until a subdivision depth of $s = 30$. The whole calculation took approximately 1 minute.

Additionally, we also tried to solve this problem with the `GLOBSOL` software. But if we use the standard parameter settings, only 13 roots were detected. Hence, `GLOBSOL` seems not to be able to handle this problem.

In this context we would also like to mention a special case of clustering which

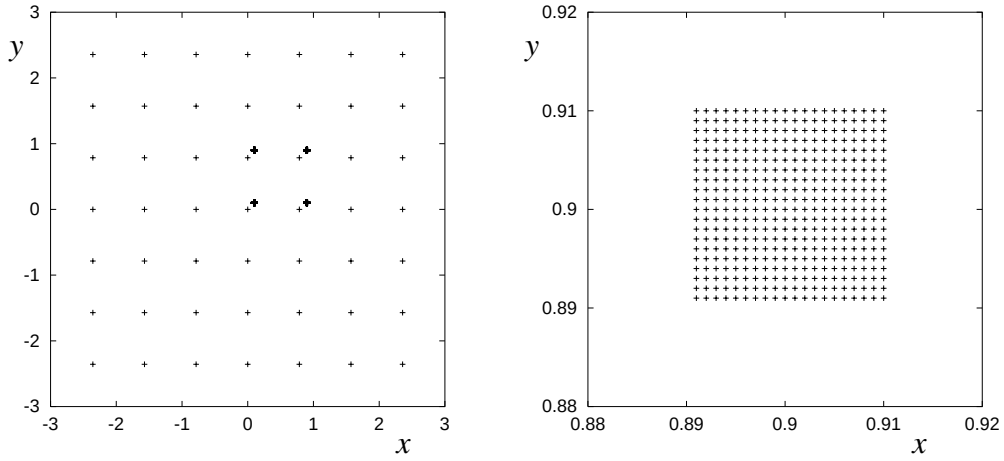


Figure 6.2: *Numerical computation of all roots for Eq. 6.19 after* 30 *subdivisions. All roots are shown in (a), and an enlargement of one of the four clusters in (b).*

might happen for our method. It occurs if one of the solutions is on the boundary of a box $M(i)$. Due to the fact that in our implementation the boundary of boxes overlap, such a periodic point then also belongs to the boundary of other boxes $M(j)$. Hence, for this and also for further subdivision steps, more than one box get subdivided for one periodic point. In Eq. 6.19 this happens for all roots which lie on the coordinate axis if we choose a covering $M = [-3.0;3.0] \times [-3.0;3.0]$. Hence, we get 1 664 boxes with roots instead of 1 649. Note, that the algorithm still converges to the same roots. This kind of clustering can be controlled with some additional tests and/or a change of the initial covering M.

Derived from Eq. 6.19, also a higher-dimensional test system is investigated by Dellnitz *et al.* [DSS02]. It is defined as follows:

$$\begin{aligned} &\mathbf{f}_{D2} : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbf{x} = (x_1, \ldots, x_d) \\ &\mathbf{f}_{D2}(\mathbf{x}) = \left(\mathbf{f}_D(x_1, x_2),\ (x_3 - 3)^2, \ldots, (x_d - d)^2 \right). \end{aligned} \tag{6.20}$$

Unfortunately, this system presents a special case for which our method can not be applied easily. Reason for this is it that the root of the function $f(\mathbf{x}) = (\mathbf{x} - k)^2$ is a minimum. So, in practice, for $e = 0$, see Sec. 6.2, an approximate convex hull of a box, however precise, does never include 0 and the box is missed. This problem could be solved if we set e to an appropriate size. But if $e > 0$ and we try to solve higher-dimensional systems then this also means that we have a high growth rate

for the boxes, and the calculation becomes ineffective. In order to prevent this, we chose the test system

$$\begin{aligned} &\mathbf{f}_{D3} : \mathbb{R}^d \to \mathbb{R}^d, \quad \mathbf{x} = (x_1, \ldots, x_d) \\ &\mathbf{f}_{D3}(\mathbf{x}) = \left(\mathbf{f}_D(x_1, x_2),\ (x_3 - 3)^3, \ldots, (x_d - d)^3 \right), \end{aligned} \tag{6.21}$$

which is similar to $\mathbf{f}_{D2}$. For $d = 5$, $M = [-3.0; 3.0]^2 \times [0.0; 6.0]^3$ and $p = 5 \cdot 2^5$ scan points per box, we converge under optimal conditions to all 1 649 roots. The calculation until $k = 14$ took 122 minutes. Except for the two examples described here, our method was also capable to detect the roots for the other examples with a dimension ≤ 6 which were presented in Dellnitz' work.

Chapter 7

Application of the RIM Method

In the last chapter we introduced the RIM method. So far, the method was only used for the detection of roots of a function $\mathbf{f}$. Now we propose some fields of application which are closer related to the investigation of dynamical systems. We already mentioned that several investigation tasks can be formulated as root finding problems. It is now our intention to provide the necessary transformations of the investigation tasks and show how the RIM method can be applied on them.

7.1 Detection of Periodic Points

A method for the localization of periodic points was already introduced in the context of symbolic analysis, see Sec. 3.3.2. Although we gave examples of a successful application in Sec. 3.6, the performance was lacking efficiency in case that there are a large number of periodic points. This is mainly due to two shortcomings. Firstly, the high effort required for the investigation of the symbolic image graph, see Remark 3.6, and, secondly, the phenomenon of clustering. For this reason, an alternative approach based on the application of the RIM method is considered. The advantage of this new approach is that it avoids the first problem at all because no graph investigations are required and it reduces the effects of the second problem, clustering, significantly.

Besides the two techniques introduced in this work, there are several other approaches for computing periodic orbits. The approach by Hansen [Han95] is based on symbolic dynamics, see e.g. [Wal91]. It is applicable to systems which can be described by a well-ordered *symbolic alphabet*. Other techniques known to the author are all based on root finding by application of a Newton-like method. The approach by Nusse and Yorke [NY97], also mentioned in Sec. 2.2, uses a

random set of points as initial values for the Newton method in order to detect periodic points. Other approaches focus only on the localization of unstable periodic orbits which are embedded in chaotic sets, see Diakonos *et. al.* [SD97], Davidchack *et al.* [DLKB01, CD05] and references therein. Hereby, the specific structural characteristics of chaotic sets are exploited in order to get a set of appropriate initial values. The application of a Newton-like method on these initial values allows then the detection of all periodic orbits up to a specified size which is only limited by computer precision. Besides that, there are also other methods for the localization of periodic orbits in specific dynamical system classes, see e.g. [BW89, SD97]. In the context of numerical case studies, see Sec. 7.1.2, our approach is compared with some of these alternative techniques.

7.1.1 Definition of the Investigation Task

Let us consider a continuous mapping $\mathbf{f} : \mathbb{R}^d \mapsto \mathbb{R}^d$ which generates a dynamical system discrete in time. Let the manifold M be a compact region in $\mathbb{R}^d$. The manifold M defines the area of investigation in which we like to detect periodic points. More precisely, we say that for a given $p \in \mathbb{N}$ we want to find the set of periodic points in M:

$$P(p) = \{\mathbf{x} \mid \mathbf{f}^{[p]}(\mathbf{x}) = \mathbf{x},\ \mathbf{x} \in M \subset \mathbb{R}^d\}. \tag{7.1}$$

To achieve this aim, we introduce the function $\mathbf{H}_p : M \mapsto \mathbb{R}^d$ as follows:

$$\mathbf{H}_p(\mathbf{x}) = \mathbf{f}^{[p]}(\mathbf{x}) - \mathbf{x}. \tag{7.2}$$

Obviously, $\mathbf{H}_p$ is continuous because $\mathbf{f}$ is continuous. Furthermore, the union of all real roots of the function $\mathbf{H}_p$ is equivalent to the set $P(p)$. This means that we can solve the problem of finding all periodic orbits for $\mathbf{f}$ in the region M, if we can detect all real roots of $\mathbf{H}_p$ in M. It is easy to see that the RIM method can be applied to find an outer covering of the roots of $\mathbf{H}_p$ and, hence, also of $P(p)$.

We also consider the subset of $\hat{P}(p)$ which only consists of points with a least period p:

$$\hat{P}(p) = \{\mathbf{x} \mid \mathbf{x} \in P(p) \text{ and } \forall p' < p : \mathbf{f}^{[p']}(\mathbf{x}) \neq \mathbf{x}\}. \tag{7.3}$$

We provide an extension of the basic root finding algorithm which allows us to localize all sets $\hat{P}(p')$ with $p' \leq p$.

Recall that the RIM method applied on $\mathbf{H}_p$, see Eq. 7.2, finds all periodic points $\subset M$ of $\mathbf{f}$ with a period p. Obviously, this includes also all points with a period $p' < p$ and

$$p \equiv 0 \mod p'.$$

Algorithm 7.1 Finding all periodic points $\leq p$ in M.

Require: M, $\mathbf{f}$, $p, n \in \mathbb{N}^+$
Ensure: $\bigcup R(C^n) \subset M$ contains all periodic points $\leq p$ of $\mathbf{f}$, and $p(M(i))$ is set

```
... { See Alg. 6.1}
for all M(i) ∈ C^s do {Detect R(C^s) ⊆ C^s}
    l ⇐ 1
    while (l ≤ p) ∧ (M(i) ∉ R(C^s)) do {Check if a box M(i) has a root for l ≤ p}
        Set H_l(x) = f^[l](x) − x
        if hasRoot(H_l, (M(i)) then {Detect if a box M(i) is subdivided}
            R(C^s) ⇐ R(C^s) ∪ {M(i)} {M(i) contains a point with a least period l}
            p(M(i)) ⇐ l {p(M(i)) is set}
        end if
        l ⇐ l + 1
    end while
    if M(i) ∉ R(C^s) then
        Delete M(i)
    end if
end for
...
```

Consider now an extension of the original algorithm of the RIM method which is given by Alg. 7.1. Hereby, the subdivision criteria is applied for every period $l = 1, \ldots, p$, and, hence, for every box $M(i)$ the smallest least period of periodic points $\mathbf{x} \in M(i)$ can be identified. We denote this smallest least period of $M(i)$ as $p(M(i))$. If a subdivision scheme is applied, then the diameter of boxes $\delta C(s) \to 0$ as the subdivision level $s \to \infty$. Each $M(i) \in C^s$ as $s \to \infty$ is an outer covering of not more than one periodic point $\mathbf{x}_p \in M(i)$. Consequently, $p(M(i))$ is then the least period of this $\mathbf{x}_p$, and an outer covering of the set $\hat{P}(p')$ can easily be constructed by selecting all boxes $M(i) \in R(C^k)$ with $p(M(i)) = p'$. So, by application of Alg. 7.1, all sets $\hat{P}(p')$ with $p' \leq p$ can be approximated.

7.1.2 Comparisons and Numerical Case Studies

We give now some examples of usage for this method. We focus on cases for which we have several coexisting unstable periodic orbits. Hereby, our approach is also compared with those of others.

Tinkerbell Map

The first example is a 2-dimensional system, the Tinkberbell map [NY97] with the following settings:

$$\begin{aligned} &\mathbf{x}(n+1) = \mathbf{f}_T(\mathbf{x}(n)), \\ &\mathbf{f}_T : \mathbb{R}^2 \to \mathbb{R}^2, \quad \mathbf{x} = (x,y)^T \\ &\mathbf{f}_T(\mathbf{x}) = \begin{pmatrix} x^2 - y^2 + 0.9x - 0.6013y \\ 2xy + 2x + 0.5y \end{pmatrix}. \end{aligned} \tag{7.4}$$

We chose this system because we can compare our results with others reported by Davidchack *et al.* [DLKB01] and Nusse *et al.* [NY97] and verify correctness.

It is our target to find for a specified period size p all sets of periodic points $\hat{P}(p')$ with $p' \leq p$, see Alg. 7.1. Note that the orbits we want to detect are unstable. We chose $M = [-2.0; 1.5] \times [-2.0; 1.5]$ as the area of investigation. The subdivision rule Υ_p which we use here divides each box $M(i)$ into 4 new boxes with the half side length of $M(i)$. As stated in Sec. 6.2, we also must set the scan points S_ε for the boxes. For comparison we choose different settings of scan points according to Tab. 7.1. Most scan points are situated on a grid on the boundary of the box. Additionally, we choose some points in the interior and set the parameter e for the approximation of the ε-reduced sets with $\varepsilon(e)$ and the covering hulls $CP(\tilde{P}(i))$. Empirically it has shown that it is most effective if the scan points lie on the boundary. Interior points are of minor importance.

Setting	p	s	Scan points per box	e	Time	Detected points
1	≤ 10	21	$3 \times 2^2 + 1 = 13$	0.15	18 min	99.8%
2	≤ 10	21	$26 \times 2^2 + 25 = 130$	-	24 min	99.5%
3	≤ 10	21	$3 \times 2^2 + 1 = 13$	-	3 min	75.9%
4	≤ 14	31	$3 \times 2^2 + 1 = 13$	0.01	114 min	99.8% ($p \leq 10$)

Table 7.1: *Tinkerbell map: Different settings for the calculation of periodic points. The parameter p describes the number of periods to detect, and s the number of subdivision steps. The scan points are separated in, first, the number of those on the boundary and, second, those in the interior. The parameter e for the approximation of the ε-reduced sets with $\varepsilon(e)$ and $CP(\tilde{P}(i))$ was only used for setting 1 and 4. We also give here the calculation time and the percentage of detected points.*

p	Analytic	Setting 1	Setting 2	Setting 3	Sym. An.	Setting 4
1	$1 \times 2 = 2$	2	2	2	6	2
2	$2 \times 1 = 2$	2	2	2	3	2
3	$3 \times 2 = 6$	6	6	6	6	6
4	$4 \times 3 = 12$	12	12	12	12	12
5	$5 \times 6 = 30$	30	30	29	71	30
6	$6 \times 9 = 54$	54	54	54	123	54
7	$7 \times 18 = 126$	126	126	118	235	126
8	$8 \times 30 = 240$	240	240	199	520	240
9	$9 \times 56 = 504$	504	504	401	1088	504
10	$10 \times 101 = 1010$	1007	1001	685	3346	1008
11	-	-	-	-	-	2055
12	-	-	-	-	-	3975
13	-	-	-	-	-	7861
14	-	-	-	-	-	14467

Table 7.2: *Tinkerbell map: The number of periodic points for period p. We present analytic results as well as the numerically computed results for the Settings* 1–4 *and for the computation based on symbolic analysis, see column* Sym. An.

For the Settings 1–3 we calculated all periodic orbits $\leq p = 10$ in order to compare them with the results reported in [DLKB01]. We chose to apply 21 subdivisions for our initial covering. According to Eq. 6.18 and

$$\delta(M) = \sqrt{2 \cdot (1.5 - (-2.0))} \approx 2.65,$$

we get for $s = 21$ the following accuracy of our results:

$$e_s \leq \delta C(s) = \frac{1}{2^{21}} \cdot \delta(M) \approx \frac{1}{2 \cdot 10^6} \cdot 2.65 \approx 10^{-6}.$$

In Tab. 7.2 we summarized the results of our calculations. We see that almost all points were detected for the settings 1 and 2. Only some points of the 10-periodic orbits are missing. Note that although not all points were detected, the method found in both cases all periodic orbits. The missing points could easily be reconstructed by forward iterations starting from the detected points. For setting 3, a larger number of periodic points is missing which is due to the fact that we chose only a small number of scan points, and no parameter e for correction. However, the computation time for setting 3 is significantly lower and still, most of the orbits could be reconstructed.

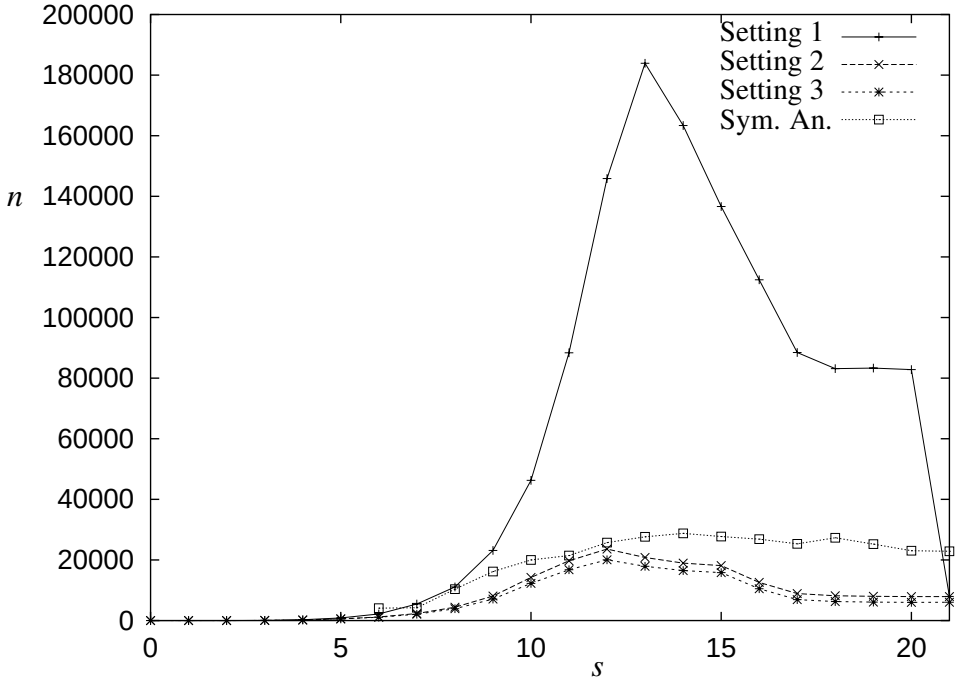


Figure 7.1: *Tinkerbell map: The number of boxes n for each subdivision step s in the calculation of the periodic points. Note that we have a strong decrease in the last subdivision for Setting* 1*. This is due to the fact, that we set $e = 0$ for the last subdivision.*

In Fig. 7.1 we reported the number of boxes for each subdivision step in the calculation of the periodic points. It is clearly to see that more than necessary boxes get subdivided. Clustering occurs and the method does not converge under optimal conditions, as would be desired. The number of boxes rather more increases for each subdivision step up to a peak around the 12-th subdivision, then it reduces and eventually converges optimally. According to our theorems, such behavior was expected and is due to the fact, that boxes are selected by the subdivision criteria which might contain roots.

For comparison, we also computed the periodic points with period ≤ 10 using the investigation method of symbolic analysis, see Sec. 3.3.2. The same settings as before are used for the area of investigation M and the subdivision of the boxes. For this technique, a lower number of scan points is sufficient. We chose to take 17 points per box. The computation takes around 4 minutes, and the results are also documented in Tab. 7.2 and Fig. 7.1. In terms of performance, the method is

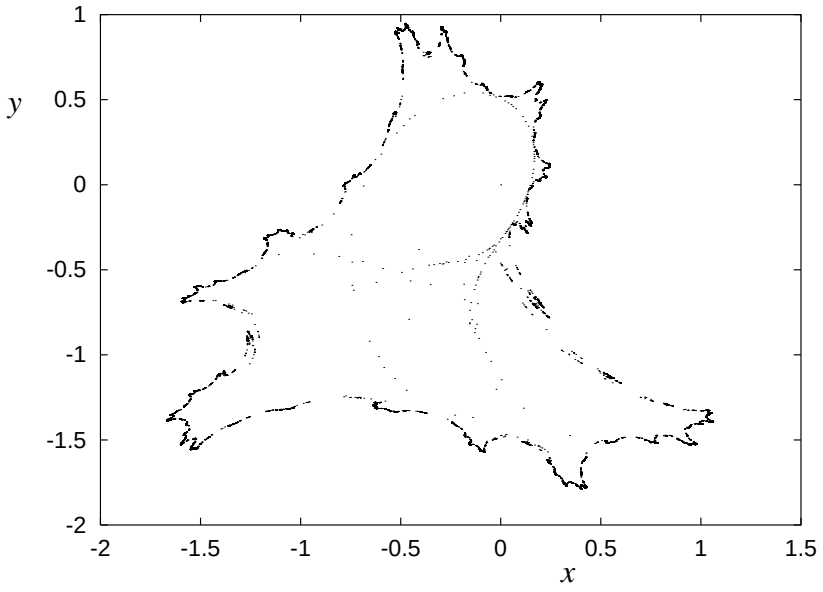


Figure 7.2: *Tinkerbell map: Numerical computation of all periodic points up to period* $p = 14$.

competitive but strong clustering can be observed. In contrast to the RIM method, the effect of clustering does also not decrease during the subdivision process but keeps constant. Hence, the precision of the computation is much lower.

Another example for a calculation is demonstrated by Setting 4. Here we calculated all periodic orbits up to $p = 14$ and applied 31 subdivisions. Note that we set the parameter $e = 0.01$ for $s = 0,\ldots,19$, and $e = 0$ for $s = 20,\ldots,31$. In Fig. 7.2 all detected points are shown. According to Tab. 7.2, the number of detected periodic points increases sharply. This leads also to an increase of computation time and stronger clustering, see Fig. 7.3. The peak for the number of boxes per subdivision is higher and only for subdivisions $s \geq 25$ we get optimal convergence. However, for this example we eventually detected 30341 periodic points with an accuracy of $\varepsilon \leq 10^{-9}$. We computed the same investigation also with the methods of symbolic analysis. The results are also illustrated in Fig. 7.3. As one can see, clustering leads to a strong growth rate of boxes. Although the peak is lower than with the RIM method, the number of boxes remains on a higher level after the peak is reached. Due to the fact that the number of boxes is generally higher than in the last computation of periods ≤ 10, the non-linear time complexity of the method, see Remark 3.6, has a stronger effect on the performance. Therefore, the computation takes around 10 hours, and is not

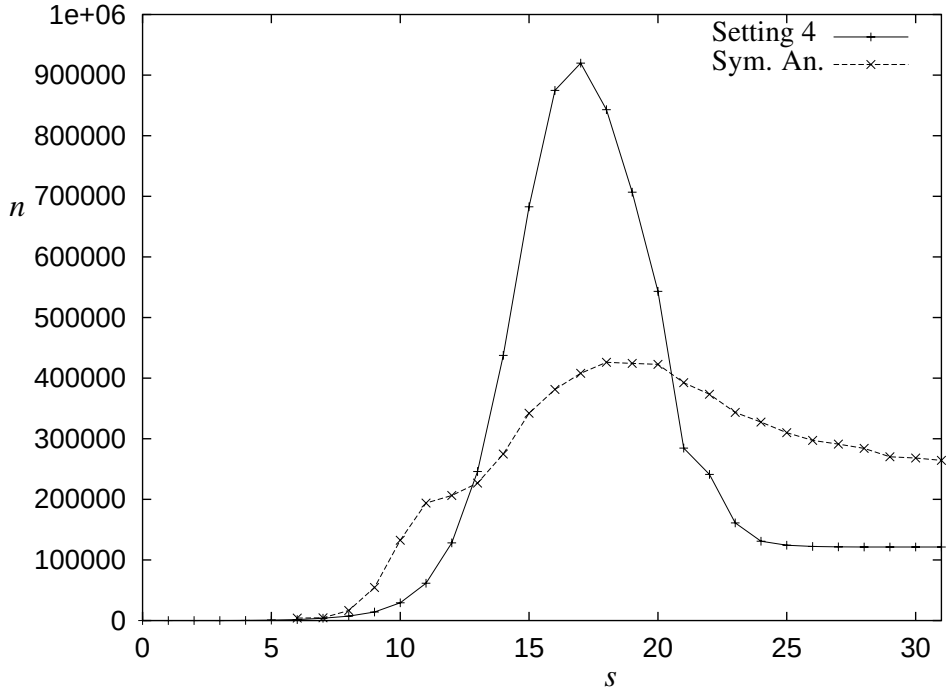


Figure 7.3: *Tinkerbell map: The number of boxes n for each subdivision step s with periodic points ≤ 14 for Setting 4 and by application of the method of symbolic analysis.*

competitive with the RIM method.

At last, we compare our results with those of others. In Nusse *et al.* [NY97], only 64 period-10 orbits were found. In this approach, the Newton method was applied on a set of random points to detect periodic points. We verified the reported results by applying the same method in our test environment. We found 87 orbits for 1 445 000 random points, and the calculation took over 20 minutes. Note that even after 40 minutes we got not more than 91 orbits. Next we look at the results reported by Davidchack *et al.* [DLKB01]. If this method is applied, all orbits up to period 10 were found, as it is the case with our method. According to the reports by Davidchack *et al.*, their method might be more efficient than ours. But it is only able to detect unstable periodic orbits embedded in chaotic sets, while our technique is applicable for all kinds of periodic orbits, as can be seen by the next example.

Discrete Food Chain Model

Now we look at a 3-dimensional system discrete in time which we already discussed earlier, namely the discrete food chain model, see Sec. 4.3.2. Note that the underlying system function $\mathbf{f}_{dfc}$, see Eq. 4.8, is not smooth. In this context we focus on a different parameter setting $\mu_0 = 4.0$, $\mu_1 = 1.0$, $\mu_2 = 3.0$ and $\mu_3 = 4.0$. At this position we can find several coexisting periodic orbits and fixed points.

Due to our numerical experiments, we strongly assume that the system has no periodic orbits with $p > 16$ for this parameter setting. Hence, we try to detect all periodic points $\leq p = 16$. The area of investigation is $M = [0;4.3] \times [0;2.1] \times [0;2.1]$. An initial covering was chosen so that the boxes have equidistant side lengths. The subdivision criteria Υ_p divides each box $M(i)$ into 8 new boxes with the half side length of $M(i)$. As in the previous example, we choose different settings for the scan points, see Tab. 7.3, but in this case all scan points are situated on the boundary of the box. Furthermore, we also do not use the e-parameter. We subdivide $s = 30$ times and, hence, get the following accuracy for our results:

$$\delta(M) \approx \sqrt{3 \cdot 2.0} \approx 2.45,$$
$$e_s \leq \delta C(s) = \frac{1}{2^{30}} \cdot \delta(M) \approx \frac{1}{2 \cdot 10^9} \cdot 2.45 \approx 10^{-9}. \quad (7.5)$$

Setting	Scan points per box	Time	Detected points
1	302	12 min	97.0%
2	158	6 min	86.4%
3	62	1 min	48.5%

Table 7.3: *Discrete food chain model: Different settings for the calculation of periodic points. The scan points are all situated on the boundary of the boxes. We also give here the calculation time and the percentage of detected points.*

For the chosen settings we get several cycles and fixed points, see Tab. 7.4 and Fig. 7.4. It is easy to check that the 4 fixed points as well as the two 8-periodic orbits and one 16-periodic orbit are unstable. The remaining two 16-periodic cycles are stable. Note that we also computed coverings for 2-periodic points. These are numerical artifacts which happened due to clustering, and all of these coverings are in the immediate neighborhood of a fixed point. For the Settings 1 and 2 we got again almost all periodic points, and for sure all periodic orbits. However, although we got by far less periodic points for Setting 3, they are still sufficient to construct all orbits by forward iteration. Except for the numerical

artifacts around one of the fixed points, each computed neighborhood for the Settings $1-3$ belongs to exactly one periodic point, and no clustering is involved.

p	Analytic	Setting 1	Setting 2	Setting 3
1	$4 \times 1 = 4$	4	4	4
2	-	5	5	1
8	$2 \times 8 = 16$	15	13	8
16	$3 \times 16 = 48$	45	40	20

Table 7.4: *Discrete food chain model: The periodic points of period p. We present analytic results as well as the numerically computed results for the Settings $1-3$. The 2-periodic points in the numerical calculation arose due to clustering.*

Next we look at the number of boxes per subdivision, see Fig. 7.5. The scenario is similar to those of the Tinkerbell map. The number of boxes increases until a threshold is reached, then shrinks and eventually converges optimally. Empirically, we can assume that this is a common scenario for the application of our method. The increase of boxes, which usually happens in the first subdivision steps, is the most crucial factor for our calculations. If it is possible to keep the growth rate low, we can detect periodic points with a high precision. If not, the computational effort might grow exponentially. Unfortunately, such a growth rate highly depends on the properties of the focused system.

We also like to mention that the investigation methods of symbolic analysis can not be applied in order to get the periodic orbits of the food chain model. Reason for this is that, due to the specific dynamics of the system, clustering occurs to a large extent. The growth rate of the symbolic image graph is exponential in every subdivision step and a distinction of the different periodic orbits can not be achieved. It turns out that the application of the RIM method is clearly advantageous for this computation.

7.2 Localization of Stable Manifolds

Another field of application for the RIM method is the computation of stable manifolds for maps. As for our other proposed method of symbolic analysis, see Sec. 5.4, neither the system's inverse nor its Jacobian is required. We are also not limited to one-dimensional manifolds. Theoretically, the method might be capable to compute manifolds independently of their dimension. In practice, we

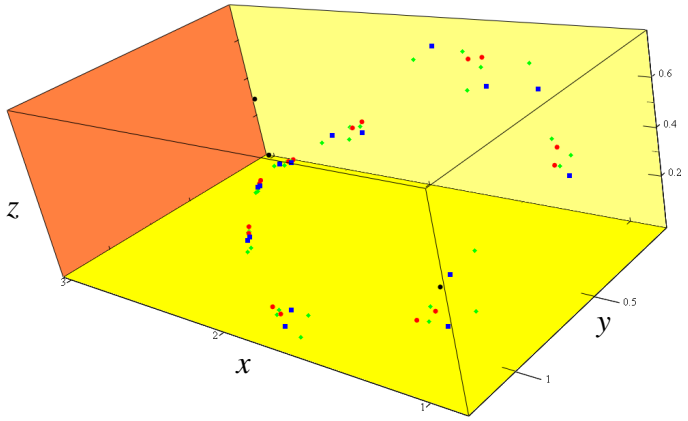


Figure 7.4: *Discrete food chain model: Numerical computation of all periodic points up to period* $p = 16$. *Fixed points (unstable) – black,* 8*-periodic points (unstable) – red,* 16*-periodic points (stable) – green,* 16*-periodic (unstable) – blue. The fixed point at the origin was neglected.*

applied it to get one- and two-dimensional manifolds.

In Sec. 2.5 several other techniques were mentioned for the same task of computation. But to our knowledge, none of these techniques is capable to compute the higher dimensional manifolds of noninvertible maps. So the advantage of the proposed method is that it can be applied in a broader range of scenarios than other techniques. Contrariwise, for invertible maps or dynamical systems continuous in time, other methods can often achieve better results concerning the performance and precision of the computation than our method.

In contrast to other approaches, the calculation of the stable manifold by the RIM method is not achieved by growth from the focused saddle, but rather more by computation of a rough approximation of the stable manifold's outer covering in a compact region $M \subset \mathbb{R}^d$. More precisely, a region M is divided into discrete sets, and for each set we decide if it covers a part of the manifold. This approach implies a limitation of the method. Only those parts of the manifold can be computed for which forward iterates reach the neighborhood of the focused saddle in less than a user-defined number p of iterations. However, although this

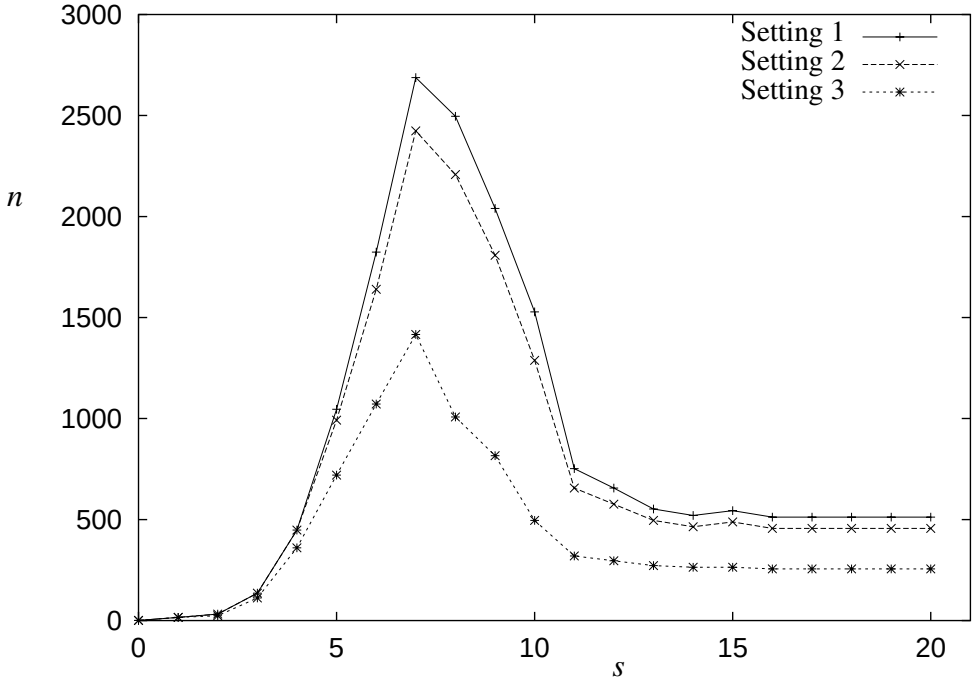


Figure 7.5: *Discrete food chain model: The number of boxes n for each subdivision step s in the calculation of the periodic points.*

is a serious drawback of the method, in many cases p can be set to a sufficiently high value for the calculation of large parts of the manifold.

In the following, we give an outline about the application of the RIM method in order to approximate global stable manifolds. Hereby, a slight variation of the original method is required for a successful computation. Furthermore, the proposed method is compared with others. Considering that the main advantage of our approach is the broad field of application, we do not focus on a comparison of performance but rather more give numerical case studies for scenarios to which other methods can not be applied to. Hereby, we mainly refer to the approach of England *et. al.* [EKO05] which allows the computation of one-dimensional stable manifolds for noninvertible and piecewise-smooth maps. We show that our method is not only applicable for the same scenarios but can also be used for the computation of higher dimensional stable manifolds for this type of dynamical systems.

7.2.1 Outline of the Method

We present a numerical method to compute an approximation of a global stable manifold $W^s(\mathbf{x}_0)$, see Eq. 2.12, for a given saddle $\mathbf{x}_0$ of a continuous map $\mathbf{f}$ within a given domain $M \subset \mathbb{R}^d$. The proposed method computes an outer approximation of a set $W^s_p(\mathbf{x}_0) \subseteq W^s(\mathbf{x}_0)$. This set is limited to a parameter p, so that a trajectory started from any $\mathbf{x} \in W^s_p(\mathbf{x}_0)$ must reach a neighborhood of $\mathbf{x}_0$ within p iterations. Later, we show that such a limitation is not as serious as it might seem on the first glance.

Recall that the RIM method can find the roots for any function. We define $\hat{\mathbf{H}}_p : M \mapsto \mathbb{R}^d$ for a $\mathbf{f}$ as

$$\hat{\mathbf{H}}_p(\mathbf{x}) = \mathbf{f}^{[p]}(\mathbf{x}) - \mathbf{x}_0, \tag{7.6}$$

whereby $\mathbf{x}_0$ is a fixed point of saddle type. Obviously, the union of all roots of $\hat{\mathbf{H}}_p$ is equivalent to all points which hit $\mathbf{x}_0$ after p iterations of $\mathbf{f}$. These points belong to the global stable manifold $W^s(\mathbf{x}_0)$. Next we consider the roots for all $\hat{\mathbf{H}}_{p'}$ with $p' \leq p$. The union of these points corresponds to all points which hit $\mathbf{x}_0$ after $p' \leq p$ iterations. More precisely, we intend to calculate for a $p \in \mathbb{N}^+$ the set

$$W^s_p(\mathbf{x}_0) = \{\mathbf{x} \in M \mid \mathbf{f}^{[k]}(\mathbf{x}) = \mathbf{x}_0 \text{ for } k \leq p\}. \tag{7.7}$$

In the following, we call $W^s_p(\mathbf{x}_0)$ the *p-limited stable set*.

Additionally, we introduce now a parameter $\tilde{\varepsilon} \in \mathbb{R}, \tilde{\varepsilon} > 0$, so that the p-limited stable set depends on $\tilde{\varepsilon}$ in a way such that

$$W^s_{p,\tilde{\varepsilon}}(\mathbf{x}_0) = \{\mathbf{x} \in M \mid \mathbf{f}^{[k]}(\mathbf{x}) \in U_{\tilde{\varepsilon}}(\mathbf{x}_0) \text{ for } k \leq p\}, \tag{7.8}$$

whereby $U_{\tilde{\varepsilon}}(\mathbf{x}_0)$ is a neighborhood of $\mathbf{x}_0$, i.e. $U_{\tilde{\varepsilon}}(\mathbf{x}_0) = \{\mathbf{x} \mid \rho(\mathbf{x}, \mathbf{x}_0) < \tilde{\varepsilon}\}$. We say that $W^s_{p,\tilde{\varepsilon}}(\mathbf{x}_0)$ is a $p,\tilde{\varepsilon}$-*limited set*. We can conclude now that

$$\lim_{\tilde{\varepsilon} \to 0} \bigcup_{p \geq 1} W^s_{p,\tilde{\varepsilon}}(\mathbf{x}_0) = W^s(\mathbf{x}_0) \cap M.$$

So we see that $\bigcup_{p \geq 1} W^s_{p,\tilde{\varepsilon}}(\mathbf{x}_0)$ converges for $\tilde{\varepsilon} \to 0$ to the global stable manifold of $\mathbf{x}_0$ in the domain M. In practice, of course, a computation of the stable manifold is limited to an approximation by a set $W^s_{p,\tilde{\varepsilon}}(\mathbf{x}_0)$ for fixed parameters p and $\tilde{\varepsilon}$. The larger p and the smaller $\tilde{\varepsilon}$ are chosen, the better is the approximation. Note that the quality of the approximation depends hereby to a large extend on the smallest stable eigenvalue $\lambda_{min} = \min(|\lambda| \mid \lambda \text{ is eigenvalue of } D\mathbf{f}(\mathbf{x}_0))$. The closer λ_{min} is to ± 1, the higher p must be set in order to compute an appropriate approximation of the stable manifold. However, we were able to approximate sufficiently large

parts of the global stable manifolds for most of the systems we tested.

Note now that for every $\tilde{\varepsilon} > 0$ and $p > 0$, an outer covering of the set $W^s_{p,\tilde{\varepsilon}}(\mathbf{x}_0)$ can be computed by application of the RIM method. Hereby it is necessary to slightly modify the original method. These modifications are subject of the following section.

7.2.2 Modifications of the Algorithm

We describe first an algorithm which computes an outer covering of a p-limited stable set. This algorithm is similar to Alg. 7.1 for the localization of periodic points. Then we will show how this algorithm can be modified to get an outer covering of a $p,\tilde{\varepsilon}$-limited stable set.

The basic algorithm for the computation of a p-limited stable set is sketched out in Alg. 7.2. The subdivision criteria is applied for every $l = 1,\ldots,p$, and, hence, every box $M(i)$ with an $\mathbf{x} \in M(i)$ so that $\mathbf{f}^{[l]}(\mathbf{x}) - \mathbf{x}_0 = 0$ belongs to $R(C^s)$. If a subdivision scheme is applied, then the diameter of boxes $\delta C(s) \to 0$ as the subdivision level $s \to \infty$. The union of all $M(i) \in C^s$ converges to $W^s_p(\mathbf{x}_0)$ as $s \to \infty$. In practice, the process of subdivision is repeated until we reach a covering $R(C^n)$ which is considered to be a good approximation of the p-limited stable set. The accuracy of the covering is decided by the largest diameter $\delta C(n)$ of the boxes $M(i) \in R(C^n)$.

The above mentioned algorithm computes a p-limited stable set. For the approximation of a global stable manifold this is usually not sufficient. Reason for this is that the computation of the p-limited set is a too strict limitation. Although trajectories $\mathbf{f}^{[p]}(\mathbf{x})$ with $\mathbf{x} \in W^s(\mathbf{x}_0)$ converge toward $\mathbf{x}_0$ for $p \to \infty$, the condition $\mathbf{f}^{[p]}(\mathbf{x}) = \mathbf{x}_0$ is usually not fulfilled for any p. Some further modifications of the algorithm are necessary. Consider therefore that the condition $\mathbf{f}^{[p]}(\mathbf{x}) \in U_{\tilde{\varepsilon}}(\mathbf{x}_0)$ can often be fulfilled for a large number of $\mathbf{x} \in W^s(\mathbf{x}_0)$. Especially in case of a numerical computation. Therefore, it is our intention to compute a $p,\tilde{\varepsilon}$-limited stable set for a small $\tilde{\varepsilon}$.

For each box $M(i)$ we consider its image $IM_l(i)$ with respect to $\hat{\mathbf{H}}_l(\mathbf{x})$, Eq. 7.6, as

$$IM_l(i) = \hat{\mathbf{H}}_l(M(i)) = \left\{ \hat{\mathbf{H}}_l(\mathbf{x}) \,|\, \mathbf{x} \in M(i) \right\}. \tag{7.9}$$

We recall that the basic idea of the RIM method is to construct an approximation of the convex hull $\mathrm{conv}(IM_l(i))$ around an image $IM_l(i)$ so that it can be decided

Algorithm 7.2 Finding a p-limited stable set $W_p^s(\mathbf{x}_0)$ for a saddle $\mathbf{x}_0$ in M.

Require: M, $\mathbf{f}$, $\mathbf{x}_0 \in \mathbb{R}^d$, $p,n \in \mathbb{N}^+$
Ensure: $\bigcup R(C^k) \subset M$ contains the p-limited stable set of $\mathbf{f}$ for $\mathbf{x}_0$
1: ... { See Alg. 6.1}
2: **for all** $M(i) \in C^s$ **do** {Detect $R(C^s) \subseteq C^s$}
3: $l \Leftarrow 1$
4: **while** $(l \leq p) \wedge (M(i) \notin R(C^s)$ **do** {Check if $M(i)$ has a root for $l \leq p$}
5: $\hat{\mathbf{H}}_l := \mathbf{f}^{[l]}(\mathbf{x}) - \mathbf{x}_0$ {Define the root finding function depending on $\mathbf{x}_0$ and l}
6: **if** $hasRoot(\hat{\mathbf{H}}_l, M(i))$ **then** {A point $\mathbf{x} \in M(i)$ hits $\mathbf{x}_0$ after l iterations}
7: $R(C^k) \Leftarrow R(C^s) \cup \{M(i)\}$ {Add $M(i)$ to $R(C^s)$}
8: **end if**
9: $l \Leftarrow l+1$
10: **end while**
11: **if** $M(i) \notin R(C^s)$ **then**
12: Delete $M(i)$
13: **end if**
14: **end for**
15: ...

if $0 \in \text{conv}(IM_l(i))$. In case $0 \in \text{conv}(IM_l(i))$ the box $M(i)$ might contain a root. Each $\mathbf{x}_{IM} \in IM_l(i)$ belongs to a neighborhood $U_{\tilde{\varepsilon}}(\mathbf{x}_0)$ for some $\tilde{\varepsilon}$. We denote by

$$\tilde{\varepsilon}_{max}(s) = \max(\rho(\mathbf{x}_{IM}, \mathbf{x}_0) | \mathbf{x}_{IM} \in IM_l(i) \in R(C^s))$$

the maximal $\tilde{\varepsilon}$ of such a neighborhood in a covering C^s. By application of a subdivision scheme, and $s \to \infty$ subdivision steps, $\tilde{\varepsilon}_{max}(s)$ goes to 0. Such a concept has been proved appropriate for the localization of roots and periodic points because $\tilde{\varepsilon}_{max}(s)$ converges relatively fast to 0, and clustering can largely be reduced. However, for the computation of a global stable manifold, the aim is to compute $\mathbf{x}_{IM} \in U_{\tilde{\varepsilon}}(\mathbf{x}_0)$ for a small $\tilde{\varepsilon}$ rather than $\mathbf{x}_{IM} = \mathbf{x}_0$. A slow convergence $\tilde{\varepsilon}_{max}(s) \to 0$ and, to some extend, also clustering is desired.

Therefore, the most significant modification of the basic algorithm relates to the approximation of an image $IM_l(i)$. Empirically, it has shown that an approximation of $IM_l(i)$ by an *axis-aligned bounding box* instead of the convex hull for $IM_l(i)$ leads to a slower convergence and better results for the approximation of stable manifolds. Such an axis-aligned bounding box is defined as

$$\text{AABB}(IM_l(i)) = [\min(IM_l(i))_1, \max(IM_l(i))_1] \times \ldots \times [\min(IM_l(i))_d, \max(IM_l(i))_d], \quad (7.10)$$

whereby $\min(IM_l(i))_k$ and $\max(IM_l(i))_k$ are the minimal and maximal value of $IM_l(i)$ on the k-th coordinate (k = 1, ..., d):

$$\min(IM_l(i))_k = \min\Big(x_k \mid \mathbf{x} \in IM_l(i) \text{ with } \mathbf{x} = (x_1,\ldots,x_d)^T\Big), \tag{7.11}$$

$$\max(IM_l(i))_k = \max\Big(x_k \mid \mathbf{x} \in IM_l(i) \text{ with } \mathbf{x} = (x_1,\ldots,x_d)^T\Big). \tag{7.12}$$

Note that a set $\mathrm{AABB}(IM_l(i))$ is convex, and that we have the relation

$$\mathrm{AABB}(IM_l(i)) \supseteq \mathrm{conv}(IM_l(i)) \supseteq IM_l(i).$$

Furthermore, for an ε-reduced set P_ε, see Def. 6.5, the bounding box $\mathrm{AABB}(CP(P_\varepsilon))$ can be computed by numerical methods and, hence, for all computations we use $\mathrm{AABB}(CP(P_\varepsilon))$ instead of $\mathrm{conv}(CP(P_\varepsilon))$, Eq. 6.9.

We can also easily decide by numerical methods if $0 \in \mathrm{AABB}(IM_l(i))$. This allows us to define a new subdivision criteria

$$\tilde{R}(C^s) = \{M(i) \,|\, M(i) \in C^s \text{ and } 0 \in \mathrm{AABB}(IM_l(i)) \text{ for a } l \in 1,\ldots,p\}. \tag{7.13}$$

It is not too difficult to prove that Theorems 6.3 and 6.4 are also valid for the subdivision criteria $\tilde{R}(C^s)$, though we converge "slower" to the solution. Due to this slower convergence rate, we were able to compute much better approximations of the global stable manifold. In Sec. 7.2.3 we will give some practical examples to show this.

Although the application of bounding boxes instead of convex hulls is usually sufficient for our purposes, we propose also a second modification for the explicit calculation of a $p,\tilde{\varepsilon}$-limited set according to Eq. 7.8. In order to do so, we recall that $d(\mathbf{x},S)$ is the distance between a point x and a set S, see Eq. 6.8. Then we can define the following subdivision criteria:

$$\hat{R}(C^s) = \big\{M(i) \,|\, M(i) \in C^s \text{ and } d\big(0, \mathrm{AABB}(IM_l(i))\big) \leq \tilde{\varepsilon} \text{ for a } l \in 1,\ldots,p\big\}.$$

Note that the numerical computation of $d\big(0, \mathrm{AABB}(I_l(i))\big)$ is an easy task. If the subdivision criteria $\hat{R}(C^s)$ is applied, we are able to compute an outer covering of a $p,\tilde{\varepsilon}$-limited set. Again, it can be shown that Theorems 6.3 and 6.4 are also valid for this subdivision criteria, and the method converges to the specified $p,\tilde{\varepsilon}$-limited set.

7.2.3 Comparisons and Numerical Case Studies

Some examples are given to demonstrate the successful application of the algorithm. We first compute some known manifolds in order to verify correctness and to compare our approach with those of others. Later we show new, previously not computed manifolds.

The Highly Interrupted Cutting Map

As a first demonstration of the method, we computed the one-dimensional stable manifold of a piecewise-smooth map. We chose the *highly interrupted cutting map* which was also used by England *et. al.*, see [EKO05] and references therein, as an example for the computation of one-dimensional stable manifolds. The system is defined as follows:

$$\begin{aligned}
&\mathbf{x}(n+1) = \mathbf{f}_{\text{hicm}}(\mathbf{x}(n)),\\
&\mathbf{f}_{\text{hicm}} : \mathbb{R}^2 \to \mathbb{R}^2, \quad \mathbf{x} = (x,y)^T \qquad (7.14)\\
&\mathbf{f}_{\text{hicm}}(\mathbf{x}) = \begin{pmatrix} A_{11}\,x + A_{12}\,y \\ A_{21}\,x + A_{22}\,y + g(x,y) \end{pmatrix}
\end{aligned}$$

with

$$\begin{aligned}
A_{11} &= \frac{e^{\varsigma\tau}}{(1-\varsigma^2)}\left(\cos\left(\sqrt{(1-\varsigma^2)}\tau\right) + \varsigma\sin\left(\sqrt{(1-\varsigma^2)}\tau\right)\right),\\
A_{12} &= \frac{e^{\varsigma\tau}}{(1-\varsigma^2)}\sin\left(\sqrt{(1-\varsigma^2)}\tau\right),\\
A_{21} &= -\frac{e^{\varsigma\tau}}{(1-\varsigma^2)}\sin\left(\sqrt{(1-\varsigma^2)}\tau\right),\\
A_{22} &= \frac{e^{\varsigma\tau}}{(1-\varsigma^2)}\left(\cos\left(\sqrt{(1-\varsigma^2)}\tau\right) - \varsigma\sin\left(\sqrt{(1-\varsigma^2)}\tau\right)\right),\\
g(x,y) &= \begin{cases} Kh^{3/4} & \text{if } h = h_0 + x - A_{11}\,x - A_{12}\,y > 0,\\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

We were able to successfully reproduce the computation of the stable manifold for a period-2 saddle

$$\begin{aligned}
P_1 &= \begin{pmatrix} 0.0876 \\ 0.5354 \end{pmatrix},\\
P_2 &= \begin{pmatrix} 0.4512 \\ 0.2983 \end{pmatrix}
\end{aligned}$$

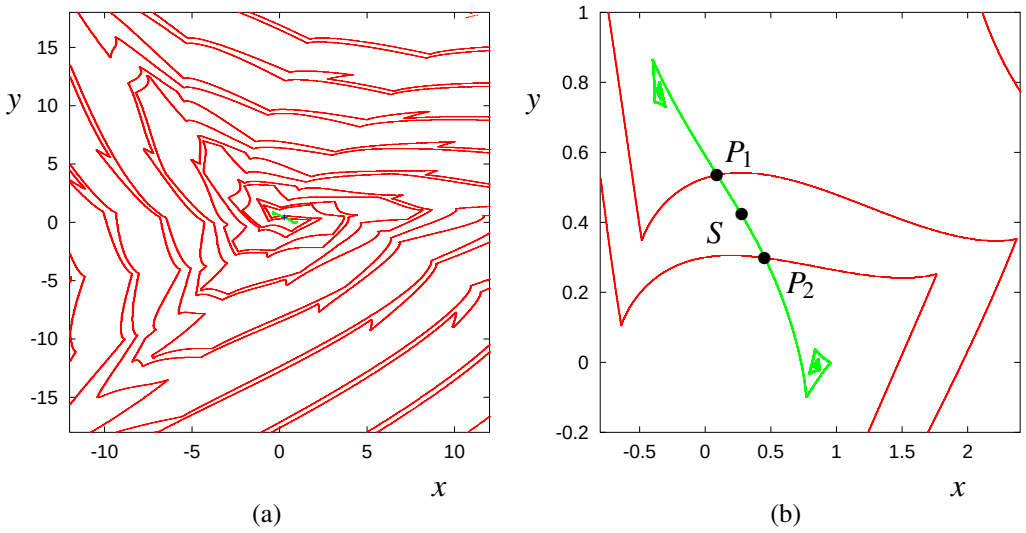


Figure 7.6: *Highly interrupted cutting map: (a) Numerical approximation of the stable manifold (red) of the period-2 saddle* $\{P_1, P_2\}$*. (b) An enlargement is given for the area where the period-2 saddle points (*P_1 *and* P_2*), the unstable manifold (green) and the sink S is situated.*

at the parameter settings $\varsigma = 0.01$, $\tau = 2$, $h_0 = 1$ and $K = 0.87$, according to [EKO05] Fig. 15. The results are shown in Fig. 7.6(a).

In order to get these results, we calculated the $p,\tilde{\varepsilon}$-limited set with $p = 30, \tilde{\varepsilon} = 0.0$ for the region $M = [-12;\ 12] \times [-18;\ 18]$. Note that an approximation of this $p,\tilde{\varepsilon}$-limited set covers the stable manifold of the whole area M. Hereby, a setting $\tilde{\varepsilon} = 0.0$ is sufficient for the numerical computation. We applied $s = 5$ subdivisions on an initial covering of 20×30 boxes. In every subdivision, a box gets divided into 2^2 new ones. The outer covering of the $p,\tilde{\varepsilon}$-limited set on the 5-th subdivision level is then given by 23406 boxes, each of these boxes has a side length of $3.75 \cdot 10^{-2}$. The total calculation time is 20 minutes, which is mainly due to the high resolution we like to acquire. For the construction of the image of each box, we took $n = 4 \times 2^2$ scan points on the boundary of the boxes. Note that a lower setting of n leads to the loss of some boxes which belong to the stable manifold's covering.

For the computation of the enlarged area around the period-2 saddle points and

the sink

$$S = \begin{pmatrix} 0.2730 \\ 0.4222 \end{pmatrix},$$

see Fig. 7.6(b), we changed the parameter settings. The area of investigation is set to $M = [-0.8; 2.4] \times [-0.2; 1.0]$, the initial covering to 20×10 boxes, and the number of scan points is reduced to $n = 2 \times 2^2$. We apply 6 subdivisions. Then the computation time reduces to 3 minutes and we get a covering of 4499 boxes with a side length $\leq 2.812 \cdot 10^{-2}$. Note that the unstable manifold was computed by the investigation method of symbolic analysis, see Sec. 5.4. Here the unstable manifold is mainly shown for illustration purposes and not subject of further analysis.

The Nien-Wicklin Map

We look now on a noninvertible map. This map was used by Nien and Wicklin [NW98] as an example for the application of an algorithm which determines the regions of the state space with different numbers of *pre-images*. The system is defined as

$$\begin{aligned} &\mathbf{x}(n+1) = \mathbf{f}_{\mathrm{NW}}(\mathbf{x}(n)), \\ &\mathbf{f}_{\mathrm{NW}} : \mathbb{R}^2 \to \mathbb{R}^2, \quad \mathbf{x} = (x, y)^T \\ &\mathbf{f}_{\mathrm{NW}}(\mathbf{x}) = \begin{pmatrix} x^4 + v\,x^2 + x\,y + \mu\,x \\ (1+a)\,y + b + ex^2 \end{pmatrix} \end{aligned} \tag{7.15}$$

Like our previous example, this map was also investigated by England *et. al.* [EKO05]. This allows us to compare the results. The main intention of England's work was it, to compute the *primary manifold* which is the unique piece of the global stable manifold $W^s(\mathbf{x}_0)$ that contains the fixed point $\mathbf{x}_0$. Furthermore, parts of the global stable manifold were calculated by using random seeds, or exploiting the special structure of the map, see [EKO05] Fig. 10. However, to our knowledge the complete global stable manifold was not yet computed.

As in the other works, we use the parameter settings $\mu = a = b = e = 0.1$ and $v = -1.9$. The map has a saddle at $P_0 = (0, -b/a) = (0, -1)$ and a period-2 repeller $\{R_1, R_2\}$ [EKO05]. We first calculated the $p, \tilde{\varepsilon}$-limited set for P_0 with $p = 15$, $\tilde{\varepsilon} = 0.0$ for the region $M = [-2.0;\ 2.0] \times [-2.2;\ 0.9]$. The result is shown in Fig. 7.7(a). Then we changed p to $p = 50$, and computed Fig. 7.7(b). Obviously, this demonstrates how p influences the part of the global stable manifold which is detected. The larger p, the larger is the detected part of the global stable manifold.

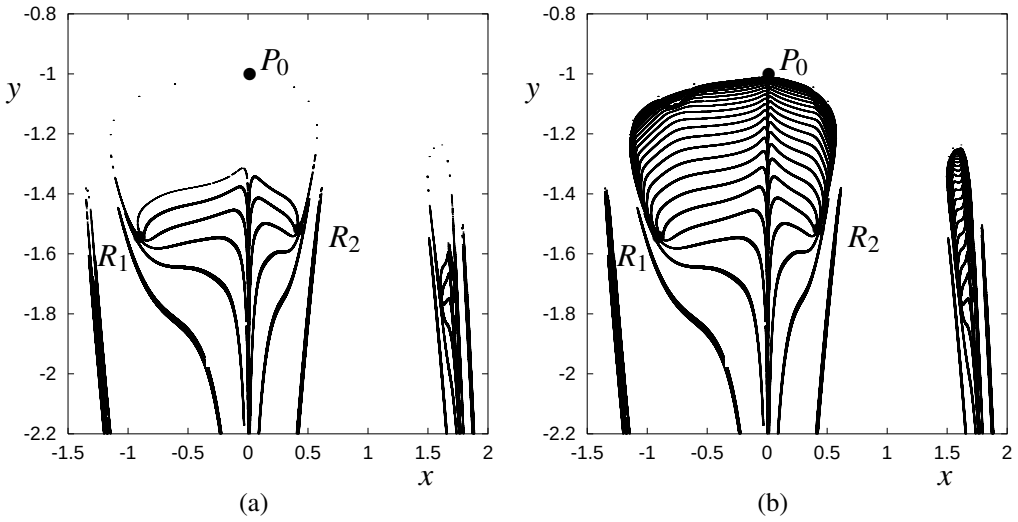


Figure 7.7: *Nien-Wicklin map: Numerical approximations of the global stable manifold $W^s(P_0)$ for the map defined by Eq. 7.15. The saddle P_0 and the period-2 repeller $\{R_1, R_2\}$ are also shown. (a) the $p,\tilde{\varepsilon}$-limited set for $p = 15$, (b) the $p,\tilde{\varepsilon}$-limited set for $p = 50$.*

For both calculations we used an initial covering of 11×7 cells and applied 9 subdivisions, whereby a box gets divided into 2^2 new ones in every subdivision. For each box we had 8×2^2 scan points. The outer coverings consist of up to $\approx 400\,000$ boxes with a side length of $\approx 8.6 \cdot 10^{-4}$. The total calculation time was 1 hour for $p = 15$ and 2 hours for $p = 50$. Note that we lost at least one box during the subdivision process. This can be concluded by observing the corner in the outer covering close to $(-0.35, -2.0)$. The loss could be avoided by increasing the number of scan points in a box.

The Nonlinear Leslie Model

After the computation of one-dimensional stable manifolds, we give now an example for the computation of a two-dimensional manifold. We chose to investigate a 3-dimensional population model. The system is based on the linear *Leslie model* [Les45] where the population is divided into d age-classes or generations. Virtually all animal and demographic forecasting models in current use are based on variations of this model. In our context, we consider a nonlinear extension of the Leslie model which was studied in [UW04] for two and three generations. We focus only on the case of three generations for which the model is defined as

follows:

$$\begin{aligned} &\mathbf{x}(n+1) = \mathbf{f}_L(\mathbf{x}(n)), \\ &\mathbf{f}_L : \mathbb{R}^3 \to \mathbb{R}^3, \quad \mathbf{x} = (x,y,z)^T \\ &\mathbf{f}_L(\mathbf{x}) = \begin{pmatrix} h(x+y+z)\cdot e^{-\lambda(x+y+z)} \\ p_1\, x \\ p_2\, y \end{pmatrix} \end{aligned} \tag{7.16}$$

This mapping is C^∞ smooth and noninvertible, whereby every point has at most two pre-images [UW04].

For our purposes, the parameter setting $h = 37.5, p_1 = 0.8, p_2 = 0.6$ and $\lambda = 0.1$ is chosen. Two coexisting attractors can be found at this position, namely a period-2 cycle

$$P = \left\{ \begin{pmatrix} 12.728 \\ 21.473 \\ 6.110 \end{pmatrix}, \begin{pmatrix} 26.841 \\ 10.183 \\ 12.884 \end{pmatrix} \right\},$$

and a four-piece chaotic attractor $\mathcal{A}$ whose shape resembles the Hénon attractor [H76]. Furthermore, there is a period-4 cycle S of saddle type

$$S = \left\{ \begin{pmatrix} 0.106 \\ 69.770 \\ 10.572 \end{pmatrix}, \begin{pmatrix} 0.968 \\ 0.084 \\ 41.862 \end{pmatrix}, \begin{pmatrix} 22.024 \\ 0.774 \\ 0.0507 \end{pmatrix}, \begin{pmatrix} 87.212 \\ 17.619 \\ 0.465 \end{pmatrix} \right\}.$$

These results were already reported in [UW04]. They can also easily be verified by the investigation methods proposed in this work. An illustration is given by Fig. 7.8. It is now our intention to approximate the stable manifold $W^s(S)$ of the period-4 saddle S.

We computed a $p,\tilde{\varepsilon}$-limited stable set with $p = 20, \tilde{\varepsilon} = 0$ for the area of investigation $M = [0;150]^3$. An initial covering of 15^3 boxes was chosen. Four subdivisions were applied, and in every subdivision, a box got divided into 2^3 new ones. As scan points of a box we used 2×2^3 points on the boundary plus the center point of the box. The result of this computation is shown in Figs. 7.9 (a) and (b). The stable manifold is approximated by an outer covering of $\approx 460\,000$ boxes with a side length $\approx 6.25 \cdot 10^{-1}$. The calculation time was 12 minutes. As expected, the stable manifold separates the domains of attraction belonging to P and $\mathcal{A}$.

We also approximated the stable manifold for a larger area of investigation $M = [-150;\ 150] \times [-150;\ 200] \times [0;\ 150]$. Setting an initial covering of

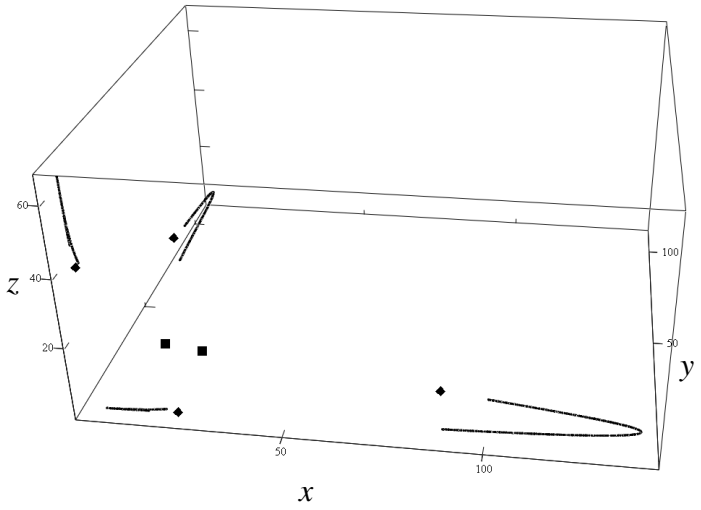


Figure 7.8: *Nonlinear Leslie Model: Numerical computation of the period-4 saddle S (marked by ♦), the period-2 cycle P (marked by ■) and the four-piece Hénon-like chaotic attractor $\mathcal{A}$.*

$35 \times 30 \times 15$ boxes, a $p,\tilde{\varepsilon}$-limited stable set with $p = 30$ was computed. The results for the subdivision level $s = 3$ are shown in Fig. 7.9(c). Hereby, the outer covering consists of $\approx 910\,000$ boxes with a side length $\approx 1.25 \cdot 10^{-1}$. The computation took around 25 minutes. Though the computation is restricted to a $p,\tilde{\varepsilon}$-limited stable set with $p = 30$, large parts of the global stable manifold can be computed. As can be seen in Fig. 7.9(c), the manifold has a highly nontrivial structure consisting of several layers which are wrapped like leaves around the attractor $\mathcal{A}$. Our method of investigation is capable of revealing these structures. Hereby, it is a distinct feature of our method that it is not required that the different parts of the manifold in the area M are connected with each other.

For comparison, we also computed the stable manifold by the methods of symbolic analysis, see Sec. 5.4. The same settings as for the first calculation were used, so that the area of investigation is $M = [0;150]^3$. In Fig. 7.10 the results of the computation after 4 subdivisions are illustrated. The results are similar to those computed by the application of the RIM method, see Figs. 7.9 (a) and (b). However, the precision of a computation by the RIM method is much higher. Although the box size is equal for both computations, the outer covering which was calculated by the method of symbolic analysis consists of $\approx 2\,300\,000$ boxes

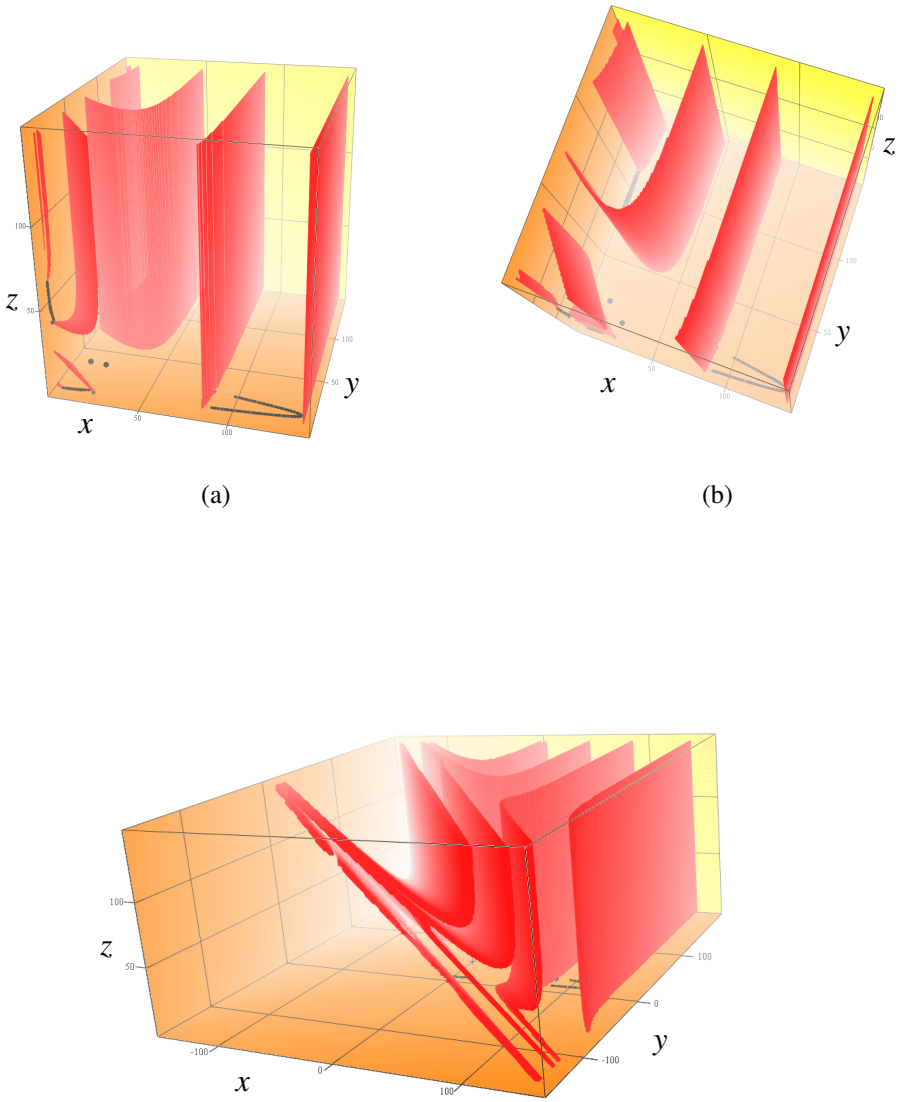
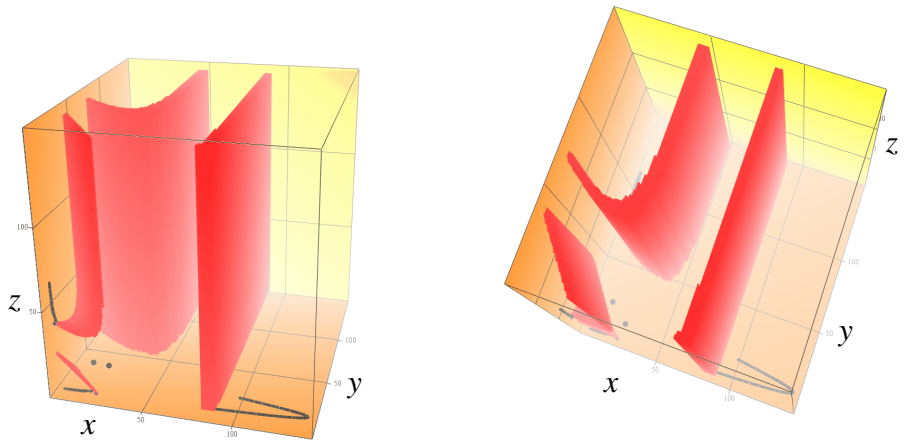


Figure 7.9: *Nonlinear Leslie Model: (a) + (b): Two different views of the outer covering of the global stable manifold $W^s(S)$ for the area of investigation $M = [0;150]^3$. The period-2 cycle P and the chaotic attractor $\mathcal{A}$ are also shown. (c) An approximation of $W^s(S)$ for the area of investigation $M = [-150;\ 150] \times [-150;\ 200] \times [0;\ 150]$.*



(a) (b)

Figure 7.10: *Nonlinear Leslie Model: (a) + (b): Two different views of a numerical approximation of the global stable manifold $W^s(S)$ using the symbolic analysis method. The period-2 cycle P and the chaotic attractor $\mathcal{A}$ are also shown.*

in contrast to the outer covering computed by the RIM method which consists of only $\approx 460\,000$ boxes, see also Fig. 7.11. In the visualization of the results, this difference is reflected by the "thickness" of the coverings. The outer covering of the stable manifold shown in Fig. 7.10 resembles rather more thick walls than 2-dimensional surfaces. Furthermore, parts of the manifold are not detected, e.g. the surface at $x > 130$. However, the numerical example we gave also shows an advantage of this method – the computation time is much lower, in case of our example it was 5 minutes in contrast to 12 minutes for the computation by the RIM method.

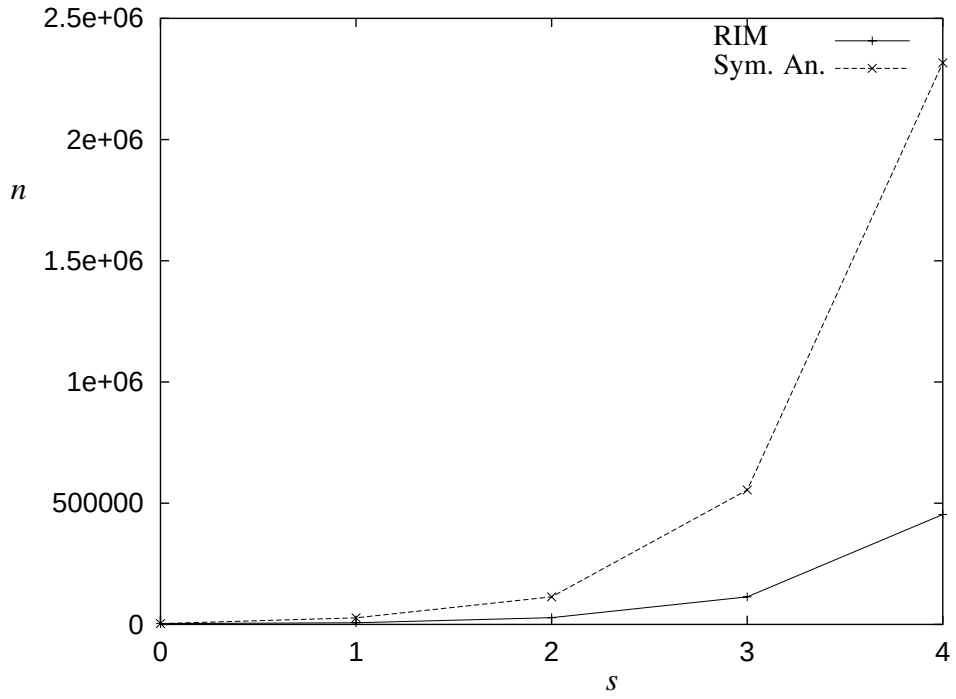


Figure 7.11: *Nonlinear Leslie Model: The number of boxes n in each subdivision step s of a numerical approximation of the global stable manifold $W^s(S)$ using the RIM method (RIM) and by application of the method of symbolic analysis (Sym. An.).*

Chapter 8

Conclusion

In this chapter we summarize the results of our work. We investigated two related approaches for the numerical investigation of dynamical systems. An overview is given about the achievements of these studies. Then follows a comparison of the two approaches with each other and with alternative strategies. Hereby, it is our intention to give a general survey about the advantages and disadvantages of our proposed methods in context to alternative strategies. The conclusion ends with an overview about open problems and fields of further research.

8.1 Achievements

Several problems regarding the numerical investigation of dynamical systems were introduced in Chapter 2. All of these problems can be solved by numerical computations based on symbolic analysis. Hereby, the investigation task is transformed into a graph problem, and then solved by analysis of the symbolic image graph. An efficient implementation of the fundamental concepts of symbolic analysis is subject of Chapter 3. Adequate data structures and algorithms are presented which allow the discretization of the phase space by a covering of boxes, the construction of a symbolic image graph for these boxes, and the subdivision of selected boxes of the covering. The most critical part of an implementation is the approximation of the covering $C(i)$, Eq. 3.3, of a box $M(i)$. We decided to choose an approximation based on scan points of a box $M(i)$. Such an approach does not allow a rigorous computation of $C(i)$ but, on the other hand, has a low computational complexity and the growth rate of the number of cells in a symbolic image is relatively low during subdivision. Further distinct features of the proposed implementation are the usage of multi-indices and a hash map for the storage of the covering. These concepts allow the efficient usage of memory resources because only those parts of the investigated area M require memory

resources which are subject of further analysis. Several numerical examples are given to illustrate the successful application of our implementation. However, also a major problem is identified which occurs for the application of every symbolic analysis-based method. This problem is clustering.

Several extensions and tunings for our implementation are proposed in Chapter 4. One of these extensions is the application of the method for dynamical systems continuous in time, i.e. an implementation of a shift operator ϕ. Several specific problems and possible solutions are discussed. Hereby, we showed that a rough approximation of the system flow is often sufficient for our investigations. On the contrary, a too precise simulation of the system flow can often hardly be computed and does not provide better results. Another extension is the introduction of an error tolerance parameter. The application of such a parameter allows a better approximation of a covering $C(i)$. As a result, the quality of the constructed symbolic image graph can be improved. Besides these extensions of the original method, we also provided some tunings for the graph construction. Namely, we introduced the application of higher function iterates and the reconstruction of fragmented solutions in order to improve the numerical computations. Both extensions are not in accordance with the original concepts of symbolic analysis but proved to be helpful in case the application of the original method fails or provides a solution of low precision due to clustering. Depending on the dynamics and dimension of the underlying system, the application of the tuning techniques can be a necessity in order to achieve appropriate results.

The construction of the symbolic image and its subdivision as discussed above provides only the basic framework for investigations on the symbolic image graph. In Section 3.3 and Chapter 5 the investigation of the graph is subject of discussion. We introduce algorithms for the realization of different investigation tasks on the symbolic image graph. The implementation of most of these tasks requires extensions and/or modifications of the theoretical concepts proposed by Osipenko [Osi04]. For this reason, we introduce the concept of the shortest periodic path which is required in order to localize periodic points. Furthermore, in the context of attractors and their basins, we make a distinction between the upper and lower bound of the domain of attraction. Based on these considerations, a different scheme for the localization of the domain of attraction is proposed. Additional theoretical considerations and proofs are also given for the construction of attractors and repellers. The theory of filtrations as proposed in [Osi99] serves as a basis for the construction but is extended. We also consider the case that the area of investigation does only cover parts of the complete area for which $\mathbf{f}$ is defined. This was neglected in the original works [Osi99] but is important for a practical application. The algorithms, which are proposed for the

realization of the theoretical concepts, have a linear time complexity and, hence, are quite efficient. We also show that these algorithms can be applied in order to fulfill other investigation tasks which were not outlined in the theoretical works of Osipenko, namely the computation of (un)stable and connecting manifolds.

Besides symbolic analysis, we propose a different framework for the investigation of dynamical systems, the so-called RIM method. The theory of the method as well as its implementation is subject of Chapter 6. Hereby, the focus is put on the core task of the method – solving the root finding problem for a large number of coexisting roots. It is our main objective to provide an approach which is on the one hand capable of detecting a large number of roots but, on the other hand, also reduces clustering to a minimum. As for symbolic analysis, the implementation of the theoretical concepts does not provide a rigorous computation of the solution. Some of the roots might get lost due to the fact that an ε-reduced set, see Def. 6.5, can only be approximated. Note, however, that in case of a rigorous computation, a larger number of boxes which do not contain a root would be considered as part of the solution and, hence, clustering would increase. Obviously, this would also impact the main objectives of the current approach. Therefore, an approximation of the solution which reduces clustering is preferred to a rigorous computation. We give numerical examples in order to show that the RIM method is still capable of detecting all or, at least, a large number of roots.

We also provide two applications of the RIM method for the investigation of dynamical systems. One of them is the detection of periodic points. Only slight changes of the basic method are necessary for such a computation. The other application is the computation of stable manifolds. Hereby, some of the computation tasks must be modified, i.e. instead of a convex hull, an axis-aligned bounding box is computed for the approximation of the image $IM(i)$ of a box $M(i)$. Furthermore, recall that a $p,\tilde{\varepsilon}$-limited set is computed. Hence, the quality of the stable manifold's approximation depends highly on the setting of p. Nevertheless, approaches towards an investigation task which are based on the RIM method are in many cases advantageous to those based on the methods of symbolic analysis. In the following section, the two different approaches are compared in detail.

8.2 Comparison of Approaches

We compare now the different approaches with each other. Detailed comparisons were already given for specific investigation tasks. In this section, the focus is put on the general characteristics of the methods. We first give a comparative survey about the advantages and disadvantages of the concepts of symbolic analysis and

the RIM method, which were both studied in this work. Then the comparison is extended to related approaches which are also based on MPSD, and to other investigation methods.

8.2.1 Symbolic Analysis and the RIM Method

As already mentioned, the RIM method was developed in order to overcome some of the shortcomings of the methods of symbolic analysis. However, the RIM method is not advantageous in every situation. It is rather more necessary to consider several aspects of the computation and of the problem which is to solve. We give a survey about some of these aspects as well as specific characteristics of the investigation methods. Note hereby that the RIM method can only be applied for the computation of periodic points and stable manifolds. Therefore, the comparison is limited to those two investigation tasks.

Performance and computational complexity For the computation of periodic points of a period $\leq p$, the computational complexity of the RIM method is clearly advantageous to those of symbolic analysis, compare Remark 3.6 and Theorem 6.6. This is especially true in case a large number of boxes n_s belongs to a covering C^s.

The situation is different for the computation of stable manifolds. In that case, the investigation of a symbolic image graph can be done in linear time, see Remark 5.4. Although the RIM method has a linear time complexity as well, the performance depends heavily on the number of function iterates. Recall that the approximation of the stable manifold by the RIM method requires the computation of a p,ε-limited set. In fact, the user-defined parameter p sets the number of function iterates which are applied for every scan point in order to compute an outer covering. Assuming the same number of boxes and scan points in the covering of a symbolic image and for a computation with the RIM method, the localization of the areas covering the solution is up to p times slower if computed with the RIM method than by symbolic analysis.

Clustering and memory consumption In several numerical case studies, see Secs. 7.1.2 and 7.2.3, it has shown that clustering can be largely reduced if the RIM method is applied instead of symbolic analysis. If clustering is reduced, then also the number of boxes in a covering is reduced and, hence, the memory consumption of the computation. However, clustering and its effects were only studied empirically. There is no theoretical proof that symbolic analysis is always affected by clustering to a larger extent than

the RIM method.

Besides the effects of clustering, also other factors are advantageous for the RIM method. In contrast to symbolic analysis, no graph is constructed for the boxes of a covering and, hence, no memory resources are required for such a graph. Furthermore, the RIM method does not require to analyze any relations between the boxes of the covering. Consequently, parts of the area of investigation M, i.e. boxes, can be investigated and subdivided independently of each other. A small part $M' \subset M$ can be investigated first. Then all boxes belonging to M' can be deleted, and the next sub-part M'' of M is investigated. As a result, the memory consumption of the RIM method is scalable.

Precision of the computation Both of our methods are not implemented for a rigorous computation. Hence, parts of the solution might get lost in the computation. Empirically, it has shown that generally a larger number of scan points are required for the RIM method in order to prevent the loss of parts of the solution. On the other hand, the precision of the computation is usually higher because the RIM method is less affected by clustering than symbolic analysis.

Partly covered solutions For the methods of symbolic analysis, it is required that all parts of the solution are in the area of investigation M. That means only those periodic points in M can be detected for which all other points of the same periodic orbit also lie in M. Similarly, only those parts of the stable manifold can be detected for which forward iterates reach the saddle $\mathbf{x}_0$ without leaving the area M. Furthermore, the saddle $\mathbf{x}_0$ of the stable manifold must also belong to M. Note, however, that these limitations can be compensated to some extent by use of higher function iterates.

In contrast, the RIM method is capable to compute such partly solutions without any restrictions. Disconnected parts of the stable manifold as well as periodic points belonging to partly covered periodic orbits can be computed in M. It is also not necessary that the saddle $\mathbf{x}_0$ lies in M. These features of the RIM method can be exploited in order to scale the memory consumption and apply parallel computation.

However, there exists also a restriction for the RIM method. Namely, the limitation to a $p, \tilde{\varepsilon}$-limited set for the approximation of the stable manifold. Only those parts of the manifold can be computed for which forward iterates

come close to the saddle $\mathbf{x}_0$ within p iterations. Such a limitation does not exist for the methods of symbolic analysis.

Classes of dynamical systems Both methods can be applied to a wide range of dynamical systems discrete in time. Noninvertible as well as piecewise-smooth system functions can be investigated. A different scenario occurs for dynamical systems continuous in time. We were not able to successfully apply the RIM method in combination with a shift operator for any of the numerical examples we tested. Our empirical studies showed that the computation of a p, ε-limited set is usually a too strict limitation. In contrast, these limitations do not exist for the computation of stable manifolds by symbolic analysis.

Parallel computing Due to the fact that the RIM method is capable to compute partly solutions without any restrictions, parallel computing is easily applicable. Sub-parts of the area of investigation M can be investigated independently of each other and, hence, also distributed for a parallel computation. In case of the methods of symbolic analysis, the application of parallel computing is more difficult. Indeed, further research would be required to do so.

The comparison of our two approaches reveals that the evaluation is multi-faceted. Depending on the scenario, the application of each of the two approaches can advantageous.

8.2.2 Related Approaches

In this section we compare our proposed investigation methods with related ones which are also based on the concepts of MPSD. One of them is the cell mapping approach [Hsu87]. In comparison with this approach, a general advantage of our methods is that only parts of the area of investigation M are investigated. Other areas, which are not further investigated, are discarded. This allows a significant reduction of memory space during the subdivision process, especially if the target object covers only a small area in M, e.g. in case of periodic points. In contrast, the cell mapping approach applies a multilevel refinement procedure and investigates always the complete area M. This might lead to a higher memory consumption. Note, however, that the strategy of refinement can be advantageous if large areas of M belong to the solution, like, for instance if the domain of attraction is computed. In that case, an adaptive subdivision scheme is still required for a further improvement of our method, see also the notes in Sec. 8.3.

Besides the computational features, there are also significant differences regarding the field of investigation. Generally, the cell mapping approach decides for each cell only if it is persistent, i.e. recurrent, or transient. By our proposed methods, several more investigation tasks can be applied, like the construction of filtrations, attractors and repellers or the localization of (un)stable and connecting manifolds. Furthermore, the investigation can be focused on selected objects, like specific attractors and their basins.

Another scheme of computation, which can probably be considered closest to ours, is the set-oriented approach by Dellnitz *et. al.* [DJ02]. Similar concepts for multilevel subdivision and discretization of the phase space are used. Also, some of the investigations tasks like localization of the chain recurrent set and invariant manifolds can be solved similarly as with the methods of symbolic analysis. Nevertheless, there exist significant differences. Firstly, the computation of stable and connecting manifolds, see [DH97, DJT01], requires the computation of the function's inverse which is not necessary in our approach. Secondly, the implementation of the basic concepts differ, as reported in Sec. 3.2.4. We also proposed tunings for the computation tasks, see Sec. 4.2, which are essential for the solution of computational problems like clustering. Thirdly, also the field of investigation differs. To our knowledge, no set-oriented approach for the construction of attractors and filtrations, the computation of basins, and periodic points was yet proposed. Also, the improvements of the RIM method to symbolic analysis, like for instance the reduction of clustering, are to some extend also improvements in comparison with the set-oriented methods.

Besides the set-oriented methods, also the computation of multivalued mappings, see [Mis02], is a concept closely related with symbolic analysis. In this approach, a symbolic image-like graph is constructed and investigated. Although some of the graph algorithms are similar to the ones we proposed in this work, they are used for other investigation tasks, mainly in the context of the Conley Index Theory. However, the incorporation of these methods of graph investigation into our framework of implementation might be a possible future task.

8.2.3 Other Investigation Methods

A significant feature of our proposed methods in comparison with others is the wide field of application. All investigations can be applied if the underlying system function is a continuous mapping. The methods of symbolic analysis are furthermore also applicable for dynamical systems continuous in time in case a shift operator can be defined. Additionally, the investigations are not

limited by the dimension of either the underlying dynamical system nor of the object which is investigated. Most of the other methods of investigation are restricted by one of these criteria. Approaches which are based on the Newton method like [NY97, DLKB01] in case of the detection of periodic points require a smooth mapping and often also a good initial guess. Several techniques for the computation of stable and connecting manifolds are restricted to invertible systems because the inverse must be computed see, for instance, [KO98a, YKY91, DJT01]. Additionally, also the application of a Newton-based iteration scheme is required for such approaches if the inverse is not available. Other approaches are limited by the dimension of the solution as in case of [EKO05] which allows only the computation of 1-dimensional stable manifolds. Several other methods are developed specifically for systems continuous in time [KOD$^+$05].

Although the methods of investigation we proposed in this work can be successfully applied to a large number of dynamical systems, the use of other investigation methods might still be advantageous. If they can be applied, methods which are not based on the concepts of MPSD often achieve better results in terms of performance, memory consumption and precision. The application of our techniques can in many cases be considered as only the first step of an investigation which produces information necessary for the application of more specialized and precise methods. Consider as an example the computation of periodic orbits. The methods proposed in this work have proved to be adequate for the localization of such orbits. But for a precise computation one should consider to apply a Newton-based method using the results produced by our methods as initial guesses.

Furthermore, although symbolic analysis and the RIM method are generally applicable to a large class of dynamical systems, not every computation produces appropriate results. Empirically, it has shown that computations might be imprecise or fail due to memory overflow, often caused by clustering. In case the dimension of the underlying dynamical system or of the object which is target of the investigation is too large, the computation can as well fail because of the lack of memory resources. Note also that an outer covering always covers parts of the investigated area M which do not belong to the solution. Such parts can be complete boxes, or even complete components of the chain recurrent set. Besides, a computation can be imprecise in the sense that parts of the solution get lost during the subdivision process or are not detected at all. Reason for this is that our implementation is not capable to compute the image of a box rigorously. Instead, such an image is approximated by a limited number of scan points. As a result, the user of our investigation methods should be critical about the computations

and use other techniques to verify the results whenever possible.

8.3 Outlook

The fields of investigation which are subject of this work are still wide open. Further research is not only required for MPSD in general but also in the areas of symbolic analysis and the RIM method. Indeed, it is our intention to present not only results of our research but also to raise new questions and propose directions for further research. We give now a survey about those aspects which we consider the most important for future research in the context of the methods of numerical investigation proposed in this work.

Further investigation methods Other investigation methods than the ones we proposed can be incorporated in the basic frameworks. In case of symbolic analysis, such investigation methods were already proposed and partly also implemented, like the computation of invariant sets [KMO03] or of the Morse Spectrum [ORAP04]. Note that the Morse Spectrum is closely related with Lyapunov exponents and can be used to verify hyperbolicity, structural stability or controllability [Osi97, Osi04]. Additionally, investigation methods based on related approaches like multivalued mappings and set-oriented methods can be incorporated into our implementation.

Some tasks might also be approached by the RIM method. We believe that there might be potential especially in the field of global optimization [Han92]. Also, some further investigation tasks for dynamical systems, like the computation of the domain of attraction, might be approached by this method.

Besides that, the development of other conceptual frameworks like symbolic analysis or the RIM method might be an area of further research.

Adaptive subdivision We have used a multilevel subdivision scheme for the computation of outer coverings. In our approach, we considered that each box of an outer covering is subdivided in a subdivision step. However, depending on the investigation task, it might not be necessary to subdivide each box. Some of the boxes might cover an area which belongs completely to the solution. A subdivision of these boxes only increases memory consumption but does not improve the precision or quality of the computation. Hence, the subdivision should be avoided for such boxes. A subdivision

scheme which subdivides only those boxes by which the quality of the computation can be increased is denoted an adaptive subdivision scheme, see also [DJ98, DJ02] for a similar approach.

In order to apply such an adaptive subdivision scheme it is required that the solution of the investigation task is an object of the same dimension than the boxes, i.e. the underlying dynamical system. Only in such a case it can happen that the complete area covered by a box can belong to the solution. Considering the investigation tasks which were subject of this work, only one of them fulfills this condition. Namely, the computation of the domain of attraction.

Other classes of dynamical systems The numerical methods we proposed can be applied for the investigation of dynamical systems whose underlying system function is a continuous mapping. Further research is required in order to find out if and how these methods can be applied to other classes of dynamical systems. For instance, numerical experiments have shown that the methods can generally also be applied to systems based on discontinuous mappings.

Also, the application on systems which are continuous time must be further investigated. It is yet unclear if and how the RIM method can be applied on these systems. Furthermore, we have seen that specific problems exist for the application of the methods based on symbolic analysis.

Higher dimensional dynamical systems The numerical case studies presented in this work were mainly restricted to two- and three-dimensional dynamical systems. Reason for this is that computations become increasingly difficult in higher dimensions. This is firstly due to the strong increase of memory consumption and, secondly, to the difficulties in visualization of the results. Note that a visualization is in many cases helpful or even required for the verification and interpretation of the computed results. The solution of these problems is a task of further investigation.

Rigorous computation We have already mentioned that no rigorous computation of an outer covering of a solution is provided by our approaches. Parts of the solution might get lost during the computation. We argued that such an approach has the advantage to reduce the effects of clustering significantly. Nevertheless, the target should still be a rigorous computation of the solution, either as an alternative method, or by a technique which is also capable to reduce clustering.

Investigation of clustering We observed that the phenomenon of clustering is one of the main obstacles for our computations. Although we proposed methods which allow the reduction of the effects caused by clustering, further research is also necessary in order to fully understand and control this phenomenon.

Optimal parameter settings The computations depend on a large number of parameter settings, like the number and size of boxes, their subdivision properties, the number of scan points and function iterates, and several specific parameters depending on the investigation method. The setting of these parameters is to a large extent experimental and requires user experience. Hereby, an optimal setting depends highly on the properties of the underlying dynamical system. Therefore, a promising area of research is the investigation of these parameter settings. A result of research could be some general guidelines, or even structured methods for the analysis of dynamical systems in order to compute such optimal settings.

Further tunings We have proposed several tunings for our computation methods. These tunings have proved to be useful tools in order to extend the area of application for our methods. However, further possibilities of tuning might be subject of research.

Parallelization We mentioned in Sec. 8.2.1 that parallelization can easily be applied to the RIM method. Parallelization might also be applied to at least some methods of symbolic analysis. For instance, different components of the chain recurrent sets could be computed independently of each other. However, experimental implementations by us have revealed that several problems occur in a practical computation which have to be considered and solved.

Our experience has shown that the capabilities of the investigation methods studied in this work depend to a large part on the solution of those problems which occur in the implementation and practical application of the theoretical concepts. These problems were also the main subject of our research. However, this field of research is still wide open, and we believe that there is a large, yet undiscovered, potential for the application of the proposed methods as well as for the development of new ones based on the concepts of MPSD.

Appendix A

Zusammenfassung

Das Thema dieser Arbeit ist die numerische Untersuchung dynamischer Systeme. Ziel ist die Entwicklung von Methoden zur Bestimmung topologischer Strukturen die von besonderer Bedeutung für die globale Analyse sind. Signifikant für die vorgestellten Verfahren ist, dass diese kein a priori Wissen über das zu untersuchende System voraussetzen und ihre Anwendung auch keinen Beschränkungen in Bezug auf die Stabilität der untersuchten Strukturen unterliegt. Zudem können die Verfahren auf ein breites Spektrum von Klassen zeitdiskreter wie auch zeitkontinuierlicher dynamischer Systeme angewandt werden.

A.1 Relevanz der Arbeit

Die Untersuchung nichtlinearer dynamischer Systeme anhand rein analytischer Methoden ist aufgrund der Komplexität dieser Systeme nur in den seltensten Fällen möglich. Daher stützt sich die moderne Theorie der dynamischen Systeme zu einem großen Teil auf die Ergebnisse numerischer Untersuchungen. Obwohl die Nutzung moderner Rechnersysteme einen erheblichen Beitrag zur Erforschung dynamischer Systeme leistet, ist die Entwicklung und Anwendung numerischer Verfahren ein bei weitem noch nicht abgeschlossenes Forschungsfeld. Grund dafür ist dass das hauptsächlich benutzte Standardverfahren, die Simulation einzelner Trajektorien, gewöhnlich nicht ausreicht um alle Merkmale der Dynamik zu erfassen. Für eine vollständige globale Analyse ist die Simulation nahezu aller Trajektorien des dynamischen Systems erforderlich. Dieser Anspruch kann im Falle einer numerischen Simulation nicht erfüllt werden. Es werden daher alternative Ansätze benötigt, die die globale Dynamik eines Systems angemessen approximieren.

Zwar existieren momentan schon einige solcher alternativer Methoden zur numerischen Untersuchung dynamischer Systeme, doch können mit diesen bei weitem nicht alle Anforderungen einer globalen Analyse abgedeckt werden. Ein wesentliches Manko ist es, dass für die Anwendung vieler der Methoden meist a priori Wissen über die zu untersuchende Struktur benötigt wird oder dass Methoden nur solche Strukturen detektieren, die ein stabiles Langzeitverhalten aufweisen. Eine Vielzahl von Methoden nutzen auch spezifische Charakteristika gewisser Systemklassen. Diese Methoden arbeiten meist sehr effizient und genau, doch ihre Anwendung ist auch auf diese speziellen Klassen beschränkt.

Die von uns vorgestellten Verfahren zielen darauf ab, die genannten Einschränkungen zu überwinden. Die Verfahren erfordern kein a priori Wissen über das zugrunde liegende System und sind auch nicht durch das Stabilitätsverhalten der Lösung eingeschränkt. Außerdem sind die Verfahren grundsätzlich auf ein breites Spektrum von Klassen dynamischer Systeme anwendbar. Dadurch eröffnet sich die Möglichkeit einer umfangreicheren globale Analyse. Insbesondere wenn noch kein Wissen über ein zu untersuchendes dynamisches System vorhanden ist, kann die Anwendung der hier vorgestellten Methoden von Bedeutung sein.

A.2 Ziele und Vorgehensweise

Ziel der Arbeit ist die Erforschung von Methoden zur Lokalisierung diverser topologischer Strukturen die von besonderer Bedeutung für die globale Analyse dynamischer Systeme sind. Bei diesen Strukturen handelt es sich um periodische Orbits, das *Chain Recurrent Set*, Repeller, Attraktoren und deren Einzugsgebiete sowie um stabile, instabile und verbindende Mannigfaltigkeiten. Für einige der Untersuchungsmethoden existieren schon vergleichbare Ansätze. In entsprechenden Fällen werden die Gründe aufgeführt, weshalb ein alternativer Ansatz gewählt wurde. Meist besteht der Vorteil der hier eingeführten Methoden in einem breiteren Anwendungsgebiet. In anderen Fällen sind unsere Verfahren aber auch zuverlässiger oder effizienter.

Mehrere Entwicklungsstufen müssen bei der Umsetzung der Berechnungsmethoden betrachtet werden. Zuerst werden die mathematischen Grundlagen entworfen. Eine mathematisch korrekte Beschreibung der Methode wie auch die Beweisführung über die Korrektheit der Berechnung werden hierbei ausgearbeitet. Anschließend muss ein entsprechender Algorithmus zur Implementierung der Methode formuliert werden. Auch die Korrektheit der Algorithmen muss nachgewiesen werden. Die praktische Anwendung erfordert es zudem, dass die

Algorithmen nicht nur korrekt sondern auch effizient arbeiten. Aus diesem Grund ist es notwendig, sowohl die Leistungsmerkmale der Methoden zu analysieren als auch Details der Implementierung, wie z.B. die verwendeten Datenstrukturen, zu betrachten. Im letzten Schritt werden dann numerische Fallstudien auf Basis einer konkreten Implementierung erstellt. Die Gesamtheit dieser Schritte ist ein interdisziplinäres Unterfangen, das sowohl Aspekte der Mathematik wie auch der Informatik berücksichtigen muss.

Es ist unsere Absicht, im Rahmen dieser Arbeit eine Übersicht über alle aufgeführten Entwicklungsstufen zu geben. Für einige der Methode wurden die mathematischen Grundlagen schon untersucht. In diesem Fall liegt der Fokus auf der Entwicklung einer korrekten und effizienten Implementierung. Für andere Methoden ist solch grundlegendes Wissen noch nicht bekannt, so dass auch die mathematisch-theoretischen Hintergründe erforscht werden müssen. Als wesentliches Resultat unserer Untersuchungen wurde eine Software entwickelt, die die praktische Anwendung der eingeführten numerischen Verfahren erlaubt. Diese Software ist Teil des nicht-kommerziellen Software-Pakets AnT [ALS+03], welches zum Download zur Verfügung steht, siehe [ant05].

A.3 Vorgestellte Untersuchungsmethoden

Das grundlegende Verfahren auf dem alle vorgestellten Berechnungsmethoden aufbauen ist die *mehrstufige Phasenraumdiskretisierung*. Hierbei werden zwei Konzepte verknüpft – die Diskretisierung des Phasenraumes und eine sukzessive Verfeinerung der diskretisierten Menge. Die Diskretisierung des Phasenraumes bedeutet, dass ein Teil des Phasenraumes, der so genannte *Untersuchungsbereich*, in eine endliche Anzahl von Mengen unterteilt wird. Anschließend werden anstatt jedes einzelnen Zustandes im Phasenraum nur diese Mengen als Gegenstand weiterer Analyse betrachtet. In Kombination mit dem zweiten angesprochenen Konzept, der sukzessiven Verfeinerung, wird eine ursprünglich sehr grobe Diskretisierung in mehreren Unterteilungsstufen zunehmend verfeinert. Im Detail bedeutet dies, dass im diskretisierten Phasenraum diejenigen Mengen ausgewählt werden, die einen Teil der Lösung enthalten. Diese Auswahl wird als Lösungsmenge bezeichnet. Die in dieser Lösungsmenge enthaltenen Mengen werden dann in kleinere Mengen unterteilt. Dieser Prozess wird solange wiederholt, bis eine Diskretisierung der Lösungsmenge von ausreichender Genauigkeit erreicht wurde.

Beispiel A.1. *Ein Beispiel für die mehrstufige Phasenraumdiskretisierung wird in Fig. A.1 aufgezeigt. Die Lösung ist durch* 3 *Punkte markiert. Der Phasenraum*

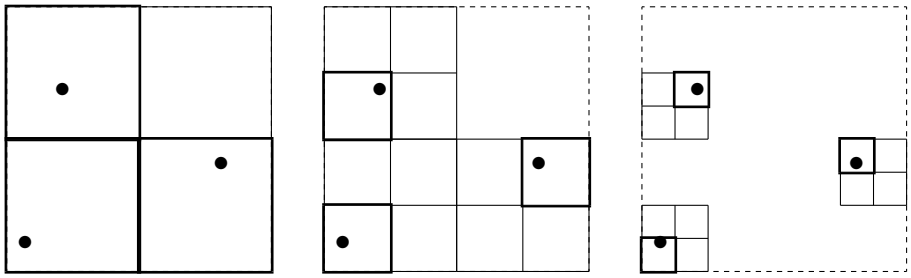


Figure A.1: Beispiel einer mehrstufigen Phasenraumdiskretisierung.

ist anfänglich in 4 *Boxen unterteilt. Diejenigen Boxen, die eine Lösung enthalten, werden für eine weitere Unterteilung ausgewählt. Diese Boxen sind in Fig. A.1 mit einem dicken Rand markiert. Nach* 3 *Unterteilungsschritten bildet die Lösungsmenge schon eine relativ genaue Approximation der Lösung.*

Auf der mehrstufigen Phasenraumdiskretisierung bauen die theoretischen Konzepte der *symbolischen Analyse* auf. Die symbolische Analyse ist ein strukturiertes Verfahren zur numerischen Bestimmung diverser Strukturen eines dynamischen Systems ohne Einschränkung in Bezug auf die Stabilität spezifischer invarianter Mengen. Die mathematischen Grundlagen wurden in einer Reihe von Arbeiten von G. S. Osipenko [Osi83, Osi93, Osi94, Osi04] vorgestellt. Das Verfahren ist nahe am *Cell-to-Cell Mapping* [Hsu87] und verwandt mit der *symbolischen Dynamik* [Wal91]. Die grundsätzliche Idee ist die Konstruktion eines gerichteten Graphen, der die Struktur des Phasenraumes für das zu untersuchende dynamische System abbildet. Diesen Graphen bezeichnet man als das *Symbolic Image* des betrachteten Systems und er kann als eine Approximation des Phasenflusses interpretiert werden. Die symbolische Analyse ist mit der mehrstufigen Phasenraumdiskretisierung dadurch verbunden, dass jeder Knoten des Symbolic Image Graphen einer Menge des diskretisierten Phasenraumes entspricht und dass eine sukzessive Verfeinerung durch eine mehrstufige Unterteilung erreicht wird.

Beispiel A.2. *Die Konstruktion des Symbolic Image Graphen wird anhand eines Beispiels illustriert (Fig. A.2). Ein Untersuchungsbereich* $M \subset \mathbb{R}^2$ *wird diskretisiert in Boxen* $C = \{M(1), \ldots, M(12)\}$. *Jede dieser Boxen entspricht genau einem Knoten im Symbolic Image Graph, siehe Fig. A.2. Sei* $\mathbf{f}$ *die Funktion, die das zu untersuchende dynamische System beschreibt. Die Fläche im Zentrum von Fig. A.2(a) ist das Bild* $\mathbf{f}(M(1))$ *der Box* $M(1)$. *Dieses überdeckt die*

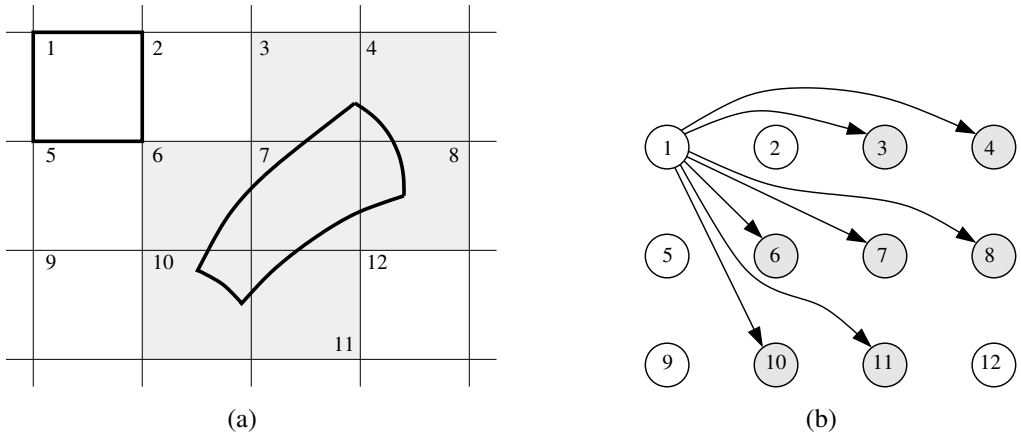


Figure A.2: *Konstruktion des Symbolic Image Graphen. (a) Die Box $M(1)$ ist durch einen dicken Rand markiert. Das Bild $\mathbf{f}(M(1))$ wird durch die Fläche im Zentrum repräsentiert. (b) Ausgehende Kanten des Symbolic Image Graph gegeben durch die Box $M(1)$.*

Boxen $\{M(3), M(4), M(6), M(7), M(8), M(10), M(11)\}$. Diese Boxen sind grau dargestellt. Der Symbolic Image Graph des dynamischen Systems besitzt dann Kanten zwischen $M(1)$ und allen überdeckten Boxen, wie in Fig. A.2(b) angezeigt. Für die komplette Konstruktion des Symbolic Image Graphen müssen nun nicht nur die ausgehenden Kanten für $M(1)$ sondern für alle Boxen $M(1) \ldots M(12)$ bestimmt werden.

Die Theorie der symbolischen Dynamik wurde detailliert beschrieben in Osipenko [Osi04]. Allerdings sind die algorithmischen Grundlagen einer Implementierung bisher nicht in ausreichendem Maße untersucht worden. In der praktischen Anwendung hat sich aber gezeigt, dass eine effiziente Implementierung von entscheidender Bedeutung für die Anwendbarkeit der symbolischen Analyse ist. Daher werden in dieser Arbeit effiziente Algorithmen und Datenstrukturen für die Konstruktion des Symbolic Image Graphen vorgestellt. Die Implementierung von Untersuchungsmethoden, die auf den Graphen angewendet werden, erfordert neben der Entwicklung von Algorithmen auch eine Erweiterung der bekannten theoretischen Konzepte. Im weiteren werden auch Ansätze zur Modifikation des Basisverfahrens aufgezeigt anhand derer das praktische Anwendungsgebiet der symbolischen Analyse stark ausgeweitet werden kann.

Obwohl die symbolische Analyse als die maßgebliche Motivation für diese Arbeit betrachtet werden kann, waren unsere Untersuchungen nicht allein auf dieses Verfahren beschränkt. In der praktischen Anwendung zeigten sich

strukturelle Nachteile bezüglich der Lösung einiger Berechnungsprobleme. Diese Nachteile konnten nicht im Rahmen der symbolischen Analyse behoben werden. Dies führte zur Entwicklung der RIM Methode, die den zweiten Teil dieser Arbeit beansprucht. Obwohl das Verfahren auch auf der mehrstufigen Phasenraumdiskretisierung aufbaut, unterscheidet es sich doch grundlegend von der symbolischen Analyse.

Im Kern befasst sich die RIM Methode mit der Lösung des Nullstellen-Problems. Dies begründet sich dadurch, dass eine Reihe von Problemen der globalen Analyse dynamischer Systeme auf solch ein Nullstellen-Problem reduziert werden kann. Hierbei liegt der Fokus auf der Lösung des Problems im Falle einer großen Anzahl koexistierender Nullstellen. Der am meisten verbreitete Ansatz für die Lösung des Nullstellen-Problems ist die Anwendung eines auf der Newton-Methode basierenden Iterations-Schemas. Dabei hat sich allerdings gezeigt, dass solch ein Ansatz einige Nachteile aufweist, die im Kontext der in dieser Arbeit vorgestellten Probleme als besonders kritisch anzusehen sind. Aus diesem Grund wird mit der RIM Methode ein alternatives Verfahren vorgeschlagen, das nicht auf der Newton-Methode basiert. Die grundlegende Idee der RIM Methode ist es, das Bild der Menge einer Diskretisierung durch eine konvexe Hülle zu approximieren. Numerische Fallstudien haben gezeigt dass die RIM Methode in einer Reihe von nicht-trivialen Szenarien bessere Ergebnisse liefert als alternative Verfahren. Wir stellen zwei Anwendungsmöglichkeiten der RIM Methode für die Untersuchung dynamischer Systeme vor. Zum Einen die Detektion periodischer Punkte. Zum Anderen die Berechnung stabiler Mannigfaltigkeiten.

Beispiel A.3. *Die grundlegende Idee der RIM Methode wird anhand eines Beispiels erläutert. Sei* $\mathbf{f}$ *die Funktion, die das zu untersuchende dynamische System beschreibt. Die Mengen* $M(i)$ *sind eine Diskretisierung des Phasenraumes. Das Bild* $\mathbf{f}(M(i))$ *einer Menge* $M(i)$ *ist in Fig. A.3(a) illustriert. Bei der RIM Methode wird dieses Bild durch eine konvexe Hülle approximiert wie in Fig. A.3(b) dargestellt. Eine Anzahl von Punkten in* $M(i)$ *wird ausgewählt. Anschließend wird eine konvexe Hülle für das Bild dieser Punkte berechnet. Diese konvexe Hülle approximiert* $\mathbf{f}(M(i))$.

Es soll angemerkt werden, dass einige weitere Verfahren existieren, die mit den von uns vorgestellten verwandt sind. In Eidenschink [Eid95] und Mischaikow [Mis02] wird ein Symbolic-Image-ähnlicher Graph, die so genannte *mehrwertige Abbildung*, konstruiert. Auch die *mengenorientierten Methoden* von Dellnitz, Hohmann und Junge [DJ02] benutzen ein ähnliches Schema wie die symbolische Dynamik und wenden eine leicht abgewandelte Unterteilungsmethode an. Hruska [Hru05] verwendet *Box Chain Construction* für

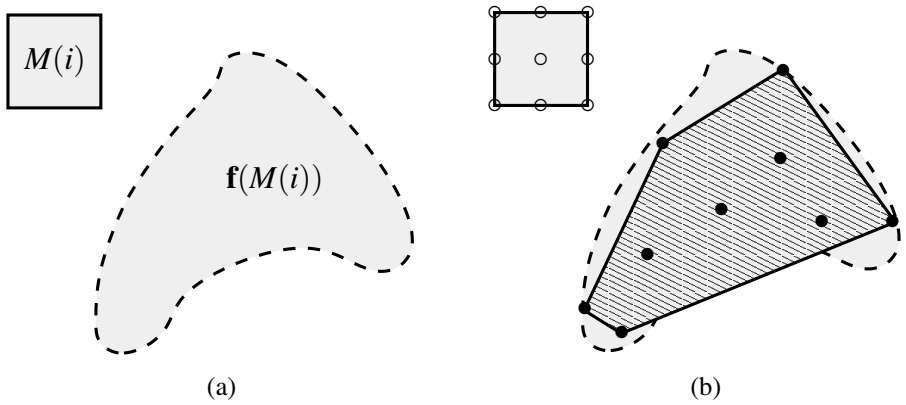


Figure A.3: *Approximation des Bildes* $\mathbf{f}(M(i))$ *einer Menge* $M(i)$. *(a) Die Menge* $M(i)$ *und deren Bild* $\mathbf{f}(M(i))$. *(b) Eine Auswahl von Punkten in* $M(i)$ *ist mit* $\circ$ *markiert. Das Bild dieser Punkte ist mit* $\bullet$ *markiert. Die konvexe Hülle ist schraffiert dargestellt.*

die Erstellung eines gerichteten Graphen mit dem Ziel, eine expandierende bzw. hyperbolische Metrik für dynamische Systeme zu berechnen. Eine genaue Abgrenzung dieser Methoden zu den von uns vorgestellten wird im Rahmen dieser Arbeit erörtert.

A.4 Praktischer Nutzen

Das Thema dieser Arbeit ist die Untersuchung dynamischer Prozesse. Bei solchen dynamischen bzw. zeitabhängigen Prozessen handelt es sich um grundsätzliche Phänomene, die in vielen Gebieten der wissenschaftlichen Forschung von Bedeutung sind. Sei es die Bewegung eines Pendels in der Physik, die Evolution einer Spezies in der Biologie oder Wechselkursschwankungen in der Wirtschaft – all dies sind zeitabhängige Prozesse. Diese Prozesse können mit den Mitteln der Mathematik beschrieben werden. Die adäquaten Werkzeuge dafür sind dynamische Systeme, die mathematischen Modelle zeitabhängiger Prozesse. Oftmals wird ein großer Aufwand betrieben, um die mathematische Repräsentation von technischen und natürlichen Prozessen zu erhalten. Dies beschränkt sich nicht nur auf die klassischen Anwendungsfelder dynamischer Systeme, den Natur- und Ingenieurswissenschaften [AFH94], sondern auch auf andere Gebiete wie die Medizin, Soziologie, Geographie oder die Wirtschaftswissenschaften.

Grund dafür ist, dass ein dynamischer Prozess besser kontrolliert und manipuliert sowie sein Verhalten vorhergesagt werden kann, wenn er durch ein dynamisches System modelliert wurde. Die Analyse des mathematischen Modells führt zu einem besseren Verständnis der zugrunde liegenden Prozesse und ermöglicht die Untersuchung des Verhaltens anhand numerischer Simulation. Die Theorie der dynamischen Systeme ist hierbei der Zweig der Mathematik der sich allgemein mit der Untersuchung dynamischer Systeme beschäftigt und daher von zentraler Bedeutung für das Verständnis dynamischer Prozesse ist.

Die im Rahmen dieser Arbeit vorgestellten numerischen Verfahren dienen nicht nur der spezifischen numerischen Untersuchung einzelner dynamischer Systeme, sondern leisten generell einen wichtigen Beitrag zur Forschung im Gebiet der Theorie nichtlinearer dynamischer Systeme. Dies begründet sich dadurch, dass der Fortschritt in diesem Forschungsfeld zu einem großen Teil auf der Beobachtung numerischer Phänomene beruht. Die Entdeckung, Verifikation wie auch Analyse derartiger Phänomene ist auf die Anwendung solcher numerischer Verfahren angewiesen, wie sie in dieser Abhandlung untersucht werden. Die vorgestellten Fallstudien sollen dies verdeutlichen und als konkrete Beispiele möglicher Anwendungsfelder dienen. Die den verwendeten dynamischen Systemen zugrunde liegenden mathematischen Modelle sind hierbei unterschiedlichen Gebieten der wissenschaftlichen Forschung entnommen, insbesondere der Geographie, Biologie, Meteorologie und der Physik. Dadurch soll die Relevanz der vorgestellten Methoden für die Wissenschaft im Allgemeinen hervorgehoben werden.

Bibliography

[ACE+87] D. Auerbach, P. Cvitanović, J.-P. Eckmann, G. H. Gunaratne, and I. Procaccia. Exploring chaotic motion through periodic orbits. *Phys. Rev. Lett.*, 58:2387–2389, 1987.

[AFH94] J. Argyris, G. Faust, and M. Haase. *An Exploration of Chaos: An Introduction for Natural Scientists and Engineers*. Elsevier Science Ltd., 1994.

[AFL+06] V. Avrutin, D. Fundinger, P. Levi, G. S. Osipenko, and M. Schanz. Investigation of dynamical systems using symbolic images: Efficient implementation and applications. *International Journal of Bifurcation and Chaos*, 16(12), 2006. To be published.

[AG90] E. L. Allgower and K. Georg. *Numerical Continuation Methods*. Springer, 1990.

[AH83] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, 1983.

[AHU87] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1987.

[Ale76] V. M. Alekseev. *Symbolic Dynamics*. 11th Mathematical School, Kiev, 1976. In Russian.

[ALS+03] V. Avrutin, R. Lammert, M. Schanz, G. Wackenhut, and G. S. Osipenko. On the software package AnT 4.669 for the investigation of dynamical systems. In G. S. Osipenko, editor, *Fourth International Conference on Tools for Mathematical Modelling*, volume 9, pages 24–35. St. Petersburg State Polytechnic University, Russia, June 2003.

[Ano67] D. V. Anosov. *Geodesic flow on closed Riemannian manifold of negative curvature*. Trudy Mathematical Steclov Institute, 1967. In Russian.

[ant05] Home page of the AnT 4.669 project, 2005. Available at `http://www.AnT4669.de`.

[BDH96] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4):469–483, 1996.

[Bey90] W.-J. Beyn. The numerical computation of connecting orbits in dynamical systems. *IMA Journal of Numerical Analysis*, 9:379–405, 1990.

[Bow82] R. Bowen. Symbolic dynamics. *Ann. Math. Soc.*, 8, 1982.

[BS70] N. P. Bathia and G. P. Szego. *Stability theory of dynamical systems*. Springer, New York, 1970.

[BW89] O. Biham and W. Wenzel. Characterization of unstable periodic orbits in chaotic attractors and repellers. *Phys. Rev. Lett.*, 63:819, 1989.

[CD05] J. J. Crofts and R. L. Davidchack. Efficient detection of periodic orbits in chaotic systems by stabilising transformations, 2005. Available at `http://arxiv.org/abs/nlin/0502013`.

[CE71] C. Conley and R. Easton. Isolated invariant set and isolating blocks. *Trans. AMS*, 158:35–61, 1971.

[CLR00] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms*. The MIT electrical engineering and computer science series. MIT Press, 2000.

[CMPY78] S. N. Chow, J. Mallet-Paret, and J. A. Yorke. Finding zero's of maps. *Math. Comp.*, 32:887–899, 1978.

[Con78] C. Conley. Isolated invariant set and the morse index. *CBMS Regional Conference Series*, 38, 1978.

[Cvi91] P. Cvitanović. Periodic orbits as the skeleton of classical and quantum chaos. *Physica D*, 51, 1991.

[Cvi92] P. Cvitanović. Focus issue on periodic orbit theory. *Chaos*, 2, 1992.

[DF89] E. J. Doedel and M. J. Friedman. Numerical computation of heteroclinic orbits. *Journal of Computational and Applied Mathematics*, 26:155–170, 1989.

[DFJ01] M. Dellnitz, G. Froyland, and O. Junge. The algorithms behind GAIO – Set oriented numerical methods for dynamical systems. In B. Fiedler, editor, *Ergodic Theories, Analysis, and Efficient Simulation of Dynamical Systems*, pages 145–175. Springer, 2001.

[DH96] M. Dellnitz and A. Hohmann. The computation of unstable manifolds using subdivision and continuation. In H. W. Broer, S. A. van Gils, I. Hoveijn, and F. Takens, editors, *Nonlinear Dynamical Systems and Chaos*, volume 19, pages 449–459. Birkhäuser, 1996.

[DH97] M. Dellnitz and A. Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numerische Mathematik*, 75:293–317, 1997.

[Dij59] W. Dijkstra. A note on two problems in connection with graphs. *Numerische Math.*, 1:269–271, 1959.

[DJ98] M. Dellnitz and O. Junge. An adaptive subdivision technique for the approximation of attractors and invariant measures. *Comput. Vis. Sci.*, 1:63–68, 1998.

[DJ02] M. Dellnitz and O. Junge. Set oriented numerical methods for dynamical systems. In G. Iooss B. Fiedler and N. Kopell, editors, *Handbook of Dynamical Systems II: Towards Applications, World Scientific*, pages 221–264, 2002.

[DJT01] M. Dellnitz, O. Junge, and B. Thiere. The numerical detection of connecting orbits. *Discrete and Continuous Dynamical Systems – Series B*, 1(1):125–135, 2001.

[DKK91a] E. J. Doedel, H. B. Keller, and J.P. Kernévez. Numerical analysis and control of bifurcation problems: I. *Int. J. Bifurcation and Chaos*, 1(3):493–520, 1991.

[DKK91b] E.J. Doedel, H.B. Keller, and J.P. Kernévez. Numerical analysis and control of bifurcation problems: II. *Int. J. Bifurcation and Chaos*, 1(4):745–772, 1991.

[DKP95] P. Diamond, P. Kloeden, and A. Pokrovskii. Cycles of spatial discretizations of shadowing dynamical systems. *Math. Nachr.*, 171:95–110, 1995.

[DLKB01] R. L. Davidchack, Y.-C. Lai, A. Klebanoff, and E. M. Bolt. Towards complete detection of unstable periodic orbits in chaotic. *Phys. Lett. A*, 287(1-2):99–104, 2001.

[DSS02] M. Dellnitz, O. Schütze, and S. Sertl. Finding zeros by multilevel subdivision techniques. *IMA Journal of Numerical Analysis*, 2(22):167–185, 2002.

[Eid95] M. Eidenschink. *Exploring Global Dynamics: A Numerical Algorithm Based on the Conley Index Theory*. PhD thesis, Georgia Institute of Technology, 1995.

[EKO05] J. P. England, B. Krauskopf, and H. M. Osinga. Computing one-dimensional stable manifolds and stable sets of planar maps without the inverse. *SIAM J. Applied Dynamical Systems*, 3(2):161–190, 2005.

[Fei78] M. J. Feigenbaum. Quantitative universality for a class of nonlinear transformations. *J. Stat. Phys.*, 19(1):25–53, 1978.

[Fei79] M. J. Feigenbaum. The universal metric properties of nonlinear transformations. *J. Stat. Phys.*, 21(6):669–707, 1979.

[Fei83] M. J. Feigenbaum. Universal behavior in nonlinear systems. *Physica D*, 7:16–39, 1983.

[Flo62] R. W. Floyd. Algorithm 97 (Shortest Path). *Communications of the ACM*, 5(6):345, 1962.

[FO03] D. Fundinger and G. S. Osipenko. Computation of attractors and their basins. In G. Osipenko, editor, *Fourth International Conference on Tools for Mathematical Modelling*, volume 9, pages 193–207. St. Petersburg State Polytechnic University, Russia, June 2003.

[Fun05] D. Fundinger. Investigating dynamics by symbolic analysis: Tunings for an efficient computation of the symbolic image. *Differential Equations and Control Processes*, (3):16–37, 2005.

[GH83] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer, N.Y., 1983.

[GK99] R. Guder and E. Kreuzer. Basin boundaries and robustness of nonlinear dynamic systems. *Archive of Applied Mechanics*, 69:569–583, 1999.

[Gol92] N. Goldenfeld. *Lectures on phase transitions and the renormalization group*. Perseus Books, 1992.

[GOY88] C. Grebogi, E. Ott, and J. A. Yorke. Unstable periodic orbits and the dimension of multifractal chaotic attractors. *Phys Rev A*, 37:1711–1724, 1988.

[GSE93] M. Gyllenberg, G. Söderbacka, and S. Ericsson. Does migration stabilize local population? Analysis of a discrete metapopulation model. *Math. Biosciences*, 118:25–49, 1993.

[GT77] S. Grossmann and S. Thomae. Invariant distributions and stationary correlation functions of one-dimentional discrete processes. *Z. Naturforsch.*, 32(a), 1977.

[GW93] J. Guckenheimer and P. Worfolk. Dynamical systems: Some computational problems. In D. Schlomiuk, editor, *Bifurcations and Periodic Orbits of Vector Fields*, pages 241–277. Kluwer Academic Publishers, 1993.

[H76] M. Hénon. A two-dimensional mapping with a strange attractor. *Commun. Math. Phys.*, 50:69–77, 1976.

[Han92] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, 1992.

[Han95] K.T. Hansen. Alternative method to find orbits in chaotic systems. *Phys. Rev. E*, 52:2388–2391, 1995.

[Has03] R. Haschke. *Bifurcations in Discrete-Time Neural Networks – Controlling Complex Network Behaviour with Inputs*. PhD thesis, University of Bielefeld, 2003.

[Hen03] M. E. Henderson. Computing invariant manifolds by integrating fat trajectories. Technical Report RC22944, IBM Research, 2003.

[HN99] W. Huyer and A. Neumaier. Global optimization by multilevel coordinate search. *J. Global Optim.*, 14:331–355, 1999.

[HOV95] A. J. Homburg, H. M. Osinga, and G. Vegter. On the computation of invariant manifolds of fixed points. *Z. Angew. Math. Phys.*, 46:171–187, 1995.

[Hru02] S. L. Hruska. *On the numerical construction of hyperbolic structures for complex dynamical systems*. PhD thesis, Cornell University, 2002.

[Hru05] S. L. Hruska. Constructing an expanding metric for dynamical systems in one complex variable. *Nonlinearity*, 18:81–100, 2005.

[Hsu80] C. S. Hsu. A theory of Cell-to-Cell mapping dynamical systems. *Journal of Applied Mechanics*, 47:931–939, 1980.

[Hsu87] C. S. Hsu. *Cell-to-Cell Mappings*. Springer, N.Y., 1987.

[HT90] R. Horst and H. Tuy. *Global Optimization*. Springer, Berlin, Germany, 1990.

[HT04] W. Huang and D. K. Tafti. A parallel adaptive mesh refinement algorithm for solving nonlinear dynamical systems. *The International Journal of High Performance Computing Applications*, 18(2):171–181, 2004.

[Hun98] F. Hunt. Unique ergodicity and the approximation of attractors and their invariant measures using Ulam's method. *Nonlinearity*, 11(2):307–317, 1998.

[Ike79] K. Ikeda. Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system. *Opt. Commun.*, 30:257–261, 1979.

[Jun99] O. Junge. *Mengenorientierte Methoden zur numerischen Analyse dynamischer Systeme*. PhD thesis, University of Paderbonn, 1999.

[Jun00] O. Junge. Rigorous discretization of subdivision techniques. In *Proceedings of Equadiff '99, Berlin*, 2000.

[Jun03] O. Junge. Computing specific isolating neighborhoods. In *Progress in analysis*, volume I,II, Berlin, 2003. World Sci. Publishing.

[Kan93] K. Kaneko, editor. *Theory and Applications of Coupled Map Lattices,*. Wiley, New York, 1993.

[kea] Home page of the GLOBSOL project. Available at `http://www.mscs.mu.edu/~globsol/`.

[Kea87] R. B. Kearfott. Some tests of generalized bisection. *ACM Trans. Math. Softw.*, 13(3):197–220, 1987.

[Kea96] R. B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer, 1996.

[Kea97] R. B. Kearfott. Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear algebraic systems. *SIAM Journal on Scientific Computing*, 18(2):574–594, 1997.

[Kle66] V. Klee. Convex polytopes and linear programming. In *Proc. IBM Sci. Comput. Symp.: Combinatorial Problems*, pages 123–158, 1966.

[KMO03] S. Y. Kobjakov, D. Y. Matiassevitch, and G. S. Osipenko. Location of the invariant set. In G. Osipenko, editor, *Fourth International Conference on Tools for Mathematical Modelling*, volume 9, pages 300–307. St. Petersburg State Polytechnic University, Russia, June 2003.

[KN90] R. B. Kearfott and M. Novoa. Algorithm 681: Intbis, a portable interval newton/bisection package. *ACM Trans. Math. Softw.*, 16(2):152–157, 1990.

[KO98a] B. Krauskopf and H. M. Osinga. Globalizing two-dimensional unstable manifolds of maps. *Int. J. Bifurcation and Chaos*, 8(3):483–503, 1998.

[KO98b] B. Krauskopf and H. M. Osinga. Growing 1D and quasi-2D unstable manifolds of maps. *J. Comput. Phys.*, 146:406–419, 1998.

[KO03] B. Krauskopf and H. M. Osinga. Computing geodesic level sets on global (un)stable manifolds of vector fields. *SIAM J. Appl. Dyn. Sys.*, 4(2):546–569, 2003.

[KOD+05] B. Krauskopf, H. M. Osinga, E. J. Doedel, M. E. Henderson, J. Guckenheimer, A. Vladimirsky, M. Dellnitz, and O. Junge. A survey of methods for computing (un)stable manifolds of vector fields. *Int. J. Bifurcation and Chaos*, 15(3):763–791, 2005.

[Les45] P. Leslie. On the use of matrices in population mathematics. *Biometrika*, 33:183–212, 1945.

[Lin02] T. Lindström. On the dynamics of discrete food chains: Low- and high-frequency behavior and optimality of chaos. *Journal of Mathematical Biology*, 45:396–418, 2002.

[LM95] D. Lind and B. Marcus. *An introduction to symbolic dynamics and coding*. New York, 1995.

[Lor63] E. N. Lorenz. Deterministic nonperiodic flow. *J. Atmos. Sci.*, 20:130–141, 1963.

[LR03] S. Lenci and G. Rega. Optimal control of nonregular dynamics in a duffing oscillator. *Nonlinear Dynamics*, 33:71–86, 2003.

[Man82] B. B. Mandelbrot. *The fractal geometry of nature*. Freeman, San Francisco, 1982.

[May76] R. M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261:459–467, 1976.

[MGBC96] C. Mira, L. Gardini, A. Barugola, and J. C. Cathala. *Chaotic Dynamics in Two-Dimensional Noninvertible Maps*, volume 20 of *World Scientific Ser. Nonlinear Sci. Ser A: Monographs and Treaties*. World Scientific, River Edge, NJ, 1996.

[Mis02] K. Mischaikow. Topological techniques for efficient rigorous computations in dynamics. *Acta Numerica*, 2002.

[MM02] K. Mischaikow and M. Mrozek. Conley index theory. In B. Fiedler, editor, *Handbook of Dynamical Systems II: Towards Applications*. North-Holland, 2002.

[MY00] J. R. Miller and J. A. Yorke. Finding all periodic orbits of maps using newton methods: Sizes of basins. *Physica D*, 135:195–211, 2000.

[nag] Home page of the NAG library. Available at `http://www.nag.com`.

[Nit71] Z. Nitecki. *Differential Dynamics*. M.I.T Press, Cambridge, 1971.

[NS75] Z. Nitecki and M. Shub. Filtrations, decompositions, and explosions. *Amer. J. of Math.*, 97(4):1029–1047, 1975.

[NW98] C.-H. Nien and F. J. Wicklin. An algoritnm for the computation of preimages in noninvertible mappings. *Int. J. of Bifurcation and Chaos*, 8(2):415–422, 1998.

[NY89] H. E. Nusse and J. A. Yorke. A procedure for finding numerical trajectories in chaotic saddles. *Phys. D*, 36:137–156, 1989.

[NY97] H. E. Nusse and J. A. Yorke. *Dynamics: Numerical Explorations*. Springer-Verlag, 1997.

[OC99] G. S. Osipenko and S. Campbell. Applied symbolic dynamics: Attractors and filtrations. *Discrete and Continous Dynamical Systems*, 5(1, 2):43–60, 1999.

[ORAP04] G. S. Osipenko, J. V. Romanovsky, N. B. Ampilova, and E. I. Petrenko. Computation of the morse spectrum. *Journal of Mathematical Sciences*, 120(2):1155–1166, 2004.

[Osi83] G. S. Osipenko. On a symbolic image of dynamical system. *Interuniv. Collect. sci. Works*, pages 101–105, 1983. In Russian.

[Osi93] G. S. Osipenko. The periodic points and symbolic dynamics. *Prog. Nonlinear Differ. Equ. Appl.*, 12:261–267, 1993.

[Osi94] G. S. Osipenko. Localization of the chain recurrent set by symbolic dynamics methods. In *Proceedings of Dynamics Systems and Applications*, volume 1, pages 227–282. Dynamic Publishers Inc., 1994.

[Osi97] G. S. Osipenko. Morse spectrum of dynamical systems and symbolic dynamics. *Proceedings of the 15th IMACS World Congress*, 1:25–30, 1997.

[Osi99] G. S. Osipenko. Construction of attractors and filtrations. *Banach center publication*, 47:173–192, 1999.

[Osi00] G. S. Osipenko. Spectrum of a dynamical system and applied symbolic dynamics. *Journal of Mathematical Analysis and Applications*, 252(2):587–616, 2000.

[Osi04] G. S. Osipenko. *Lectures on symbolic analysis of dynamical systems*. St. Petersburg State Polytechnic University, 2004.

[PC89] T. S. Parker and L. O. Chua. *Practical Numerical Algorithms for Chaotic Systems*. Springer-Verlag, Berlin, 1989.

[Pil99] P. Pilarczyk. Computer assisted method for proving existence of periodic orbits. *TMNA*, 13(2):365–377, 1999.

[Pin96] J. D. Pintér. *Global Optimization in Action*. Kluwer, Dordrecht, 1996.

[RT71] D. Ruelle and F. Takens. On the nature of turbulence. *Communications of Mathematical Physics*, 20:167–192, 1971.

[SD97] P. Schmelcher and F. K. Diakonos. Detecting unstable periodic orbits of chaotic dynamical systems. *Phys. Rev. Lett.*, 78:4733–4736, 1997.

[Sed93] R. Sedgewick. *Algorithms in Modula 3*. Addison-Wesley, Massachusetts, 1993.

[Shu87] M. Shub. *Global stability of Dynamical Systems*. Springer-Verlag, 1987.

[Sim89] C. Simó. On the analytical and numerical approximation of invariant manifolds. In D. Benest and C. Froeschlé, editors, *Les Méthodes Modernes de la Mécanique Céleste*. Goutelas, 1989.

[Spa73] C. Sparrow. *The Lorenz Equations: Bifurcations, Chaos, and Strange Attractors*. Springer, N.Y., 1973.

[Tar72] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput*, 1:146–160, 1972.

[Tar91] R. Tarjan. *Data structures and network algorithms*. Society for Industrial and Applied Mathematics. Philadelphia, Pa., 1991.

[TG88] B. H. Tongue and K. Gu. Interpolated cell mapping of dynamical systems. *ASME Journal of Applied Mechanics*, 55:461–466, 1988.

[UW04] I. Ugarcovici and H. Weiss. Chaotic systems of a nonlinearity density dependent population model. *Nonlinearity*, 17:1689–1711, 2004.

[VH94] J. Verschelde and A. Haegemans. Homotopies for solving polynomial systems within a bounded domain. *Theoretical Comp. Sci. A*, 133(3):165–185, 1994.

[Wal91] P. Walters, editor. *Symbolic dynamics and its applications*. American Mathematical Society, July 1991.

[YK89] X. Ying and N. Katz. A simple reliable solver for all the roots of a nonlinear function in a given domain. *Computing*, 41:317–333, 1989.

[YKY91] Z. You, E. J. Kostelich, and J. A. Yorke. Calculating stable and unstable manifolds. *Internat. J. Bifur. Chaos Appl. Sci. Eng.*, 1(3):605–623, 1991.

[Zie95] G. M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, 1995.

Glossary

Λ	an attractor, 20
Λ^*	a repeller, 21
$\rho(\mathbf{x},\mathbf{y})$	a ρ-norm distance between $\mathbf{x}$ and $\mathbf{y}$, 13
ζ	$\{H_k\}$, 32
Υ_p	a subdivision rule for the RIM method, 131
$AP(p)$	set of periodic points with a period $\leq p$, 15
C	a finite covering of an area M or $\mathcal{M}$ by closed sets, 30
$C(i)$	the covering of boxes $M(j) \in C$ whose intersections with $\mathbf{f}(M(i))$ are not empty, 30
c_i	the cell of a graph G which matches to a box $M(i)$ of C, 30
$\tilde{c}_i$	a cell of the dag graph representation of G, 92
c_∞	sink for all edges which leave a covering C, 32
$\mathcal{CL}(H_p)$	the connections between H_p and its larger sets $\mathcal{L}(H_p)$, 103
$\mathcal{CS}(H_p,H_q)$	the connections between a set H_p and distinct sets $H_q \in \mathcal{L}(H_p)$, 103
$D(L)$	the basin of an attractor L on G, 90
$d(c_i)$	the corresponding dag cell $\tilde{c}_k$ for a cell c_i, 93
DG	the dag graph representation of G, 92
$D_u(L)$	the upper bound of the basin of an attractor L on G, 95
$DV(G)$	the set of the dag cells corresponding to non-recurrent cells in G, 93
$DV(\zeta)$	the set of those dag cells representing a set of equivalent recurrent cells, 93

$E(\tilde{c}_i)$	adjacency list of all target cells of $\tilde{c}_i$, 93
$En(L)$	entry of an attractor L on G, 90
$Ex(L)$	exit of an attractor L on G, 90
$\mathbf{f}$	a continuous mapping which generates the dynamical system in focus, 12
$\mathbf{f}(M(i))$	the image of a box $M(i)$, 30
G	a symbolic image, 30
H_k	a subset of equivalent recurrent cells in $RV(G)$, 31
$\mathbf{H}_p$	transformation of $\mathbf{f}$ so that the roots of $\mathbf{H}_p$ are the p-periodic points of $\mathbf{f}$, 154
I	unique multi-index which defines the position of a box $M(I)$ in M, 35
$IM(i), IM$	the image $f(M(i))$ of a box $M(i)$, 133
L	an attractor on a symbolic image G, 90
$\mathcal{L}(H_p)$	the larger sets of H_p, 103
$L(S_k)$	the set $L(S_k) = \bigcup_{H_j \in S_k} L(H_j)$, 108
$\mathcal{M}$	C^∞-smooth manifold which is a compact in $\mathbb{R}^d$ and $\mathbf{f} : \mathcal{M} \mapsto \mathcal{M}$, 11
M	the area of investigation, $M \subseteq \mathcal{M}$, 30
$M(i)$	a box of a covering C, 30
$M(\infty)$	the area $\mathcal{M} \setminus C$, 32
P_ε	an ε-reduced set of IM, 134
$P(\tilde{c}_i)$	adjacency list of all parent cells of $\tilde{c}_i$, 93
$P(p)$	set of periodic points with a period p, 15
$\hat{P}(p)$	set of periodic points with a least period p, 15
$R(C^s)$	a subdivision criteria for the RIM method, 131
RO_f	the roots of $\mathbf{f}$, 130
$RV(G)$	the subset of recurrent cells $RV(G) \subseteq V(G)$, 31
$C^s, G^s, I^s, \ldots$	a covering, symbolic image, index, etc. at subdivision level s, 42

$S(I)$	the scan points of a box $M(I)$ belonging to a symbolic image, 37
S_k	recursively build selection of recurrent sets $H_k \subset \zeta$, 108
$\tilde{S}_k$	recursively build selection of recurrent sets $H_k \subset \zeta$, 109
$SV(G)$	a selection of cells $SV(G) \subseteq V(G)$ considered for subdivision, 42
$\mathbb{T}$	a time space, where $\mathbb{T} \in \{\mathbb{Z}, \mathbb{Z}_+, \mathbb{R}, \mathbb{R}_+\}$, 11
$V(G)$	the set of cells on G, 30
$W^s(\Lambda)$	the basin of an attractor Λ, 20
$W^s(\mathbf{x}_0)$	the stable manifold of a saddle $\mathbf{x}_0$, 24
$W^s_p(\mathbf{x}_0)$	p-limited stable set of a saddle $\mathbf{x}_0$, 165
$W^s_{p,\tilde{\varepsilon}}(\mathbf{x}_0)$	$p,\tilde{\varepsilon}$-limited stable set of a saddle $\mathbf{x}_0$, 165
$W^u(\mathbf{x}_0)$	the unstable manifold of a saddle $\mathbf{x}_0$, 24
$\mathbb{Z}^T$	a discrete time space, where $\mathbb{Z}^T \in \{\mathbb{Z}, \mathbb{Z}_+\}$, 12

Index