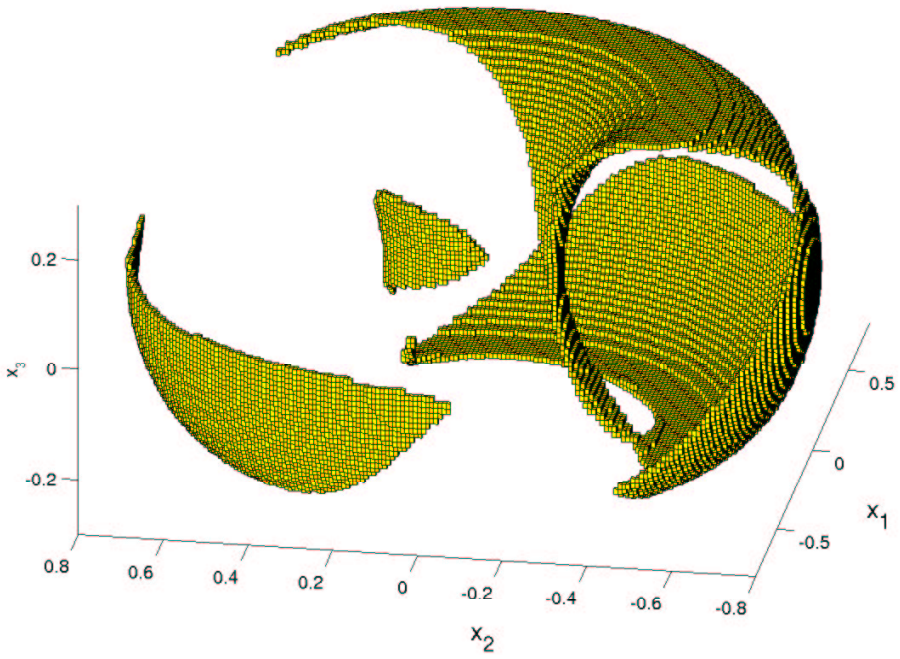
Set Oriented Methods for Global Optimization



Dissertation

von

Oliver Schütze

Schriftliche Arbeit zur Erlangung des Grades
eines Doktors der Naturwissenschaften

Fakultät für Elektrotechnik, Informatik und Mathematik
Universität Paderborn

Paderborn, 21. Dezember 2004

Gutachter:

- Prof. Dr. Michael Dellnitz
- Prof. Dr. Kalyanmoy Deb
- Prof. Dr. Björn Schmalfuss

Tag der mündlichen Prüfung: 21. Dezember 2004

Meinen Eltern

Danksagung

Die vorliegende Arbeit entstand während meiner Zeit als wissenschaftlicher Mitarbeiter der Universität Paderborn. Dort arbeitete ich an dem Lehrstuhl für Angewandte Mathematik unter der Leitung von Prof. Dr. Michael Dellnitz. Ich möchte mich an dieser Stelle für seine intensive Betreuung und die Ermöglichung der Durchführung der Arbeit bedanken. Besonders hervorzuheben ist seine Geduld, die er aufbrachte, um mir viele Zusammenhänge zwischen der Optimierung und dem Gebiet der Dynamischen Systeme zu erklären.

Mein Dank gilt ausserdem Prof. Dr. Kalyanmoy Deb und Prof. Dr. Björn Schmallfuss für die Begutachtung dieser Arbeit.

Meinen Kollegen, insbesondere Katrin Witting, Alessandro Dell'Aere, Dr. Gary Froyland, Dr. Oliver Junge und Stefan Sertl, danke ich für deren Hilfe und Unterstützung.

Ferner möchte ich der Abteilung CT PP 2 der Siemens AG unter der Leitung von Prof. Dr. Albert Gilg und dem Lehrstuhl für Regelungstechnik und Mechatronik unter der Leitung von Prof. Dr. Joachim Lückel danken, die mich stets mit relevanten Beispielen vor allem aus den Bereichen der Stabilitätsanalyse und der Mehrzieloptimierung versorgt haben.

Speziellen Dank möchte ich Dr. Qinghua Zheng aussprechen, der diese Arbeit durch zahlreiche fruchtbare Diskussionen und Hinweise auf interessante Problemstellungen entscheidend prägte.

Nicht zuletzt gilt mein Dank Kathrine Badham-Thornhill, Kathrin Padberg und Mirko Hessel, die diese Arbeit Korrektur gelesen haben.

Diese Arbeit wurde unterstützt durch die DFG-Sonderforschungsbereiche 376 "Massive Parallelität" und 614 "Selbstoptimierende Systeme des Maschinenbaus".

Contents

Chapter 1

Introduction

The personal improvement is an inherent desire of every individual. The search for the extremes is one of the biggest sources of motivation and inspiration for athletes, scientists, mathematicians and the rest of the human race who seek the ultimate solution in their subject. Since the beginning *optimization* has been a very active field in mathematics though a thorough and beautiful theory was developed only in the 1950s when computers became available. Both new generations of computers with rapidly growing capacities and new problems arising from ever advancing applications, call perpetually for new optimization methods. The scope of this thesis is to give a contribution to that issue, to develop new techniques for the solution of modern optimization problems.
To be more precise, we propose in this work numerous schemes for the numerical treatment of some global optimization problems, such as the (global) root finding problem and multi–objective optimization. Most of the algorithms which are presented here are based on a set–oriented multi–level scheme. These *subdivision techniques* can be described quite well by a well–known albeit frivolous formulation of the essence of optimization. This says that the task is to find a black cat in a dark room in minimal time (and a constrained optimization problem belongs to a room full of furniture). The general approach of the basic subdivision scheme for the computation of the "cat–finding problem" reads as follows:
The algorithm starts with a division of the complete room into a finite set of smaller and disjoint room segments. In the next step all of the segments in which the cat (or any part of it) is *not* located[1] are discarded. Continual subdivision and selection steps of the remaining parts of the room lead to a sequence of outer approximations which increasingly narrow down the requested location of the cat until it is pinpointed.

To go into the detail of this subject the following approach is used: since the approach of the subdivision algorithm is global (recall that we start with a partition of the entire room in the first step in order to locate the black cat), we will consequently address the corresponding *global* optimization problems.
In particular, in the subsequent chapters we will propose numerical algorithms for

[1] The effective realization of this part is in fact the crucial factor for the success of the algorithm.

the solution of the following problems:

- global zero finding for
  - differentiable functions $g : \mathbb{R}^n \to \mathbb{R}^n$, and for
  - analytic functions $g : \mathbb{C} \to \mathbb{C}$, due to its particular structure.
- location of stability regions of parameter dependent delay differential equations, and
- multi–objective optimization.

We propose several adaptive algorithms for the location of zeros within a prescribed compact region in $\mathbb{R}^n$ (and $\mathbb{C}$ respectively), and will demonstrate their strength by several numerical results (see Chapters 3 and 4).
Furthermore, we address the problem of the location of the stability regions of parameter dependent delay differential equations (Chapter 5). The proposed algorithm uses the stability criterion on the underlying characteristic functions. Due to its set–oriented approach, the method has the advantage over most other existing schemes that it does not require a particular structure of these functions.
The main part of this thesis consists of the numerical schemes for the computation of the set of solutions for multi–objective optimization problems (Chapter 6). We will propose algorithms for different assumptions of smoothness of the underlying models. The main result of this thesis are algorithms for objectives which are twice continuously differentiable. These set–oriented continuation–like methods can – as a "by–product" – be used for the computation of general implicitly defined manifolds, even in higher dimensional space. This allows for the efficient computation of optimization models with equality constraints (one example is shown in Figure 1.3).
One interesting fact about the subdivision techniques is that they provide a lot of potential to be parallelized efficiently. However, this is not a topic of this thesis. For particular parallelization strategies of subdivision algorithms for the analysis of general dynamical systems as well as for global optimization we refer to [109] and [110].
Finally, it has to be mentioned that parts of this thesis grew out of the publications [106], [30], [31], [29], [108], [107] and [94], for each of which the author has given a substantial contribution.

A detailed outline of this thesis is summarized as follows:
In **Chapter 2** we present a global set–oriented multi–level scheme which serves as the basis for most of the algorithms presented in this thesis. The primary variant of this scheme was proposed in [26]. It is based on a subdivision technique which allows the construction of a covering of the so–called *relative global attractor* of a given dynamical system $f : Q \subset \mathbb{R}^n \to \mathbb{R}^n$ in a given compact domain $Q \subset \mathbb{R}^n$ up to a prescribed accuracy. The relative global attractor is important for our considerations because it contains in particular all invariant sets of f within Q (a set $A \subset Q$ is called invariant with respect to f if $f(A) = A$). This fact will be utilized for the

construction of the dynamical system for the underlying optimization problem: all dynamical systems which are used in combination with the subdivision algorithms have in common that the set of interest – e.g. the minimizer of an objective function in the context of scalar optimization – is contained in the invariant set.
We extend the basic algorithm to the situation where one is interested in the computation of the *common* invariant sets of a finite number of different dynamical systems. Moreover, we prove convergence of this abstract multi–level algorithm (Section 2.2), discuss its implementation (Section 2.3) and make some estimates of the computational effort (Section 2.4). The latter discussion will show that the subdivision techniques are restricted to moderate dimensions n of the parameter space of the dynamical system.

In **Chapter 3** we develop a new method for the location of *all* roots of a given differentiable function $g : \mathbb{R}^n \to \mathbb{R}^n$ within a prescribed (compact) domain. The underlying idea is to view iteration schemes such as Newton's method as dynamical systems and then to apply the subdivision techniques mentioned above. It will evolve that it is typically preferable to work with iteration schemes with variable step sizes such as the damped Newton method since this makes the computations both more reliable and more efficient. Since these iteration schemes with different step sizes can be viewed as different dynamical systems we use the results of Chapter 2 to construct suitable algorithms. This will lead to an adaptive scheme which uses the Armijo step size (Section 3.2). We present some numerical results and compare them to a different global zero finding procedure (Section 3.3).
Prominent alternative ways to attack the global zero finding problem are mainly based on *interval analysis* (see e.g. [1], [54] or [69]) or on *homotopy methods* (for this we refer e.g. to [13], [2], [81] and [119]). The algorithms which use interval analysis are mostly rigorous but typically just applicable in the case where the dimension n of the parameter space is not too large. Although homotopy methods are in general non-rigorous, they can in certain cases be used to find the entire zero set, for instance when the function g is polynomial-like.
Finally, we propose a new method for the numerical treatment of (unconstrained) scalar optimization problems (Section 3.4). To be more precise, we combine the subdivision techniques for the location of zeros of the gradient of the objective with classical branch&bound strategies (see [62]) in order to combine the advantages of both methods. The algorithm is designed for general optimization problems and does not require particular properties of the underlying models. However, on lack of this "specialization", the approach may have problems for higher dimensional models. An alternative approach which is also based on subdivision techniques and allows the computation of certain "real world" problems is presented in [111, 12].

Chapter 4 deals with the problem of finding all the roots of an *analytic function* $g : Q \subset \mathbb{C} \to \mathbb{C}$ within a rectangle $Q \subset \mathbb{C}$. The basic idea for the algorithms proposed in this chapter is to numerically use the *argument principle* and to combine this with a multi–level subdivision strategy. This way we construct tight coverings of the set $\mathcal{N}_Q$ of zeros of g within Q. In combination with classical iteration schemes

this leads to an adaptive scheme for the approximation of $\mathcal{N}_Q$, the so–called $QZ40$ algorithm (Section 4.3). We conclude the chapter with three examples (Section 4.4) indicating the robustness and efficiency of this numerical approach. The main example (see Section 4.4.3 and also Figure 1.1) arises in the stability analyis of an annular combustion chamber, which is joint work with Siemens, Munich.
The underlying idea of the approach, namely to use the argument principle in combination with a subdivision procedure, already occurs in [121, 122] and [73]. All the algorithms including the one presented here mainly differ in the numerical realization. Furthermore, the methods developed in [91] and [120] are also similar in spirit, where degree theory is the underlying concept for the approximation of the set of zeros of a given function.

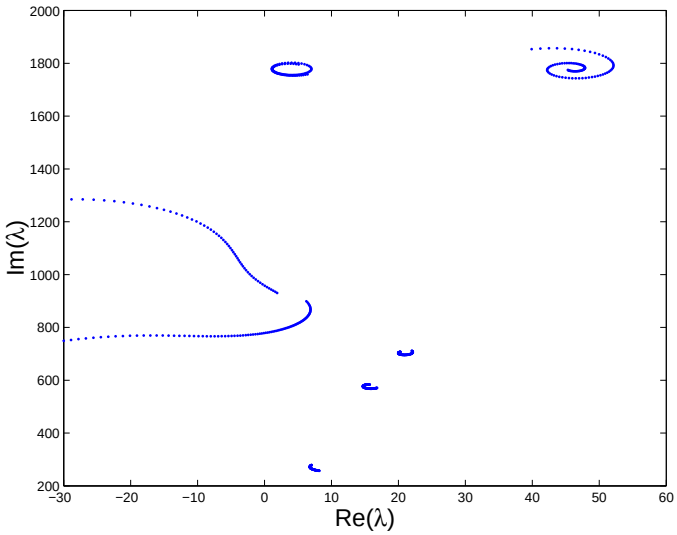


Figure 1.1: Roots of the characteristic function of a model of an annular combustion chamber under variation of a critical parameter. See Section 4.4.3 for further information.

In **Chapter 5** we address the problem of detecting the stability domain of parameter dependend delay differential equations (DDEs) which arose in an application presented in Chapter 4. Similarly to the preceeding problem classes we also propose in this context an adaptive subdivision algorithm for the outer approximation of these sets (Section 5.3), which utilizes the stability criterion on the characteristic function of the underlying DDE. To verify this criterion numerically, the root finding algorithm $QZ40$ which is presented in Chapter 4 is applied. This algorithm is applicable to a broad class of DDEs since it does not take a particular structure of the underlying model into account. In fact, the method can also be applied more generally to arbitrary bounded subsets $\mathcal{S} \subset Q$ of a given domain $Q \subset \mathbb{R}^n$ in parameter space, where $\mathcal{S}$ has to have positive Lebesque measure. Finally we close the chapter by presenting some approximations of stability regions belonging to DDEs with two and three free parameters (Section 5.4 and also Figure 1.2).
The most common methods for the computation of the stability regions which use

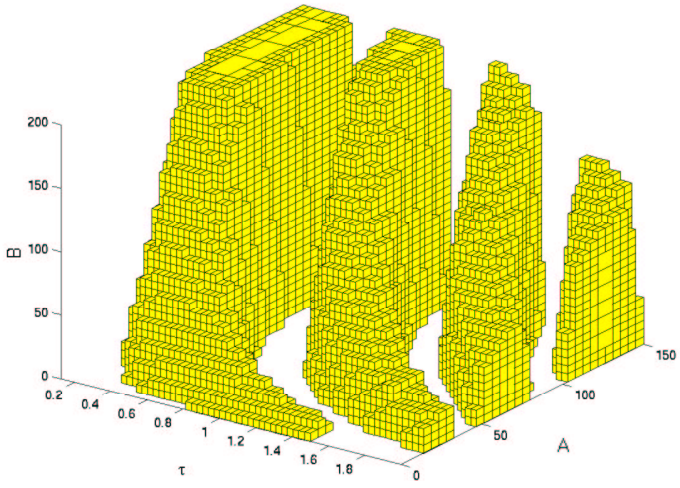


Figure 1.2: Approximation of the stability region of a parameter dependent delay differential equation with three free parameters (see Section 5.4.3).

the characteristic function are based on the works of Nyquist ([88]), Pontryagin ([93]) and Neimark ([86]). Common to these different methods is that they are very effective on a particular class of DDEs but none of them can be used generally.

Chapter 6 deals with the numerical treatment of *multi–objective optimization problems* (MOPs). Here, the main task is the construction of algorithms for the approximation of the entire set of solutions of a given MOP, the so–called *Pareto set*. We start with the development of three basic algorithms (Section 6.3) which can be applied when all objectives of the underlying optimization model are differentiable. In addition to two subdivision algorithms (i.e. DS–Subdivision and the Sampling Algorithm) we also propose a further technique: the class of *Recovering* algorithms can be viewed as a (set–oriented) variant of a continuation method. These algorithms are local in nature, but on the other hand are not restricted to moderate dimensions like the algorithms based on subdivision techniques. A best overall performance – i.e. a global and robust method which can also attack higher dimensional problems – can be achieved by a combination of these three algorithms which will be explained in Section 6.3.4.
After having stated the basic algorithms we refine them for MOPs with different smoothness assumptions. To be more precise, we go into details for MOPs where the objectives are not differentiable as well as for optimization models where all objectives are twice continuously differentiable.
In the case the model consists of non–differentiable objectives, combinations of *Multi–Objective Evolutionary Algorithms* (MOEAs, see e.g. [125], [22], [23] or [20]) with both the subdivision and the recovering techniques are proposed (Section 6.6). For smooth MOPs it turns out that the performance of the recovering techniques can be improved drastically (Section 6.8). The boxes which are used for the realization of the algorithm serve as a surprisingly effective storage tool for the representation

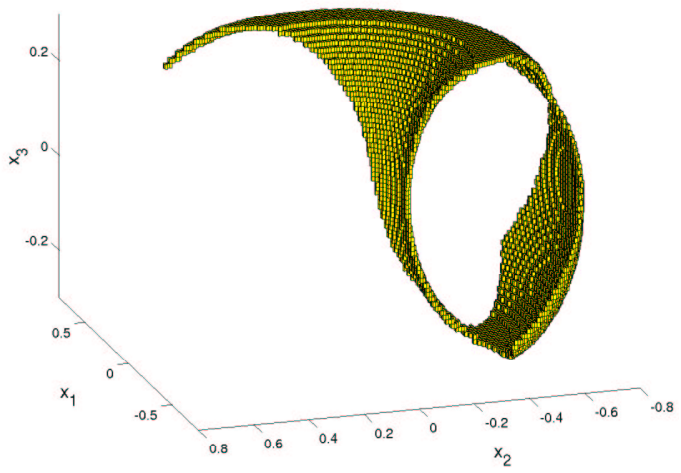


Figure 1.3: Pareto set of a multi–objective optimization problem containing an equality constraint (see Section 6.9.3).

of the solution set, in particular in higher dimensions. Since this technique can also be used for the computation of general implicitly defined manifolds, also MOPs with equality constraints can be treated effectively (see Figure 1.3). The recovering techniques for smooth MOPs can be viewed as an improvement of the homotopy approach which is described in [58], since the algorithm presented in that work is only applicable locally and does not treat adequately the case where the MOP has more than two objectives.
Numerous numerical results of non–differentiable, differentiable and twice continously differentiable MOPs taken from literature and emerging in applications will be given in Sections 6.7, 6.4 and 6.9.
In addition, a new tree–based data structure for the effective computation of the *nondominated sorting problem* which occurs in many algorithms for the treatment of MOPs – e.g. the Sampling Algorithm presented in the same chapter – will be proposed (Section 6.5). To attack this problem, there exist the intuitive linear list approach, the *quad tree* approach (see [52], [117], [116] and [84]), and the *composite point* approach ([39]) which have all the average case complexity of $O(n^2)$ for vector comparisons – including the newly presented approach. We conclude this section with a comparison of the new method to the quad tree approach and the linear list approach by a category of problems.

We finally close this thesis with a conclusion in **Chapter 7**.

Chapter 2

The Subdivision Algorithm

2.1 Introduction

In this chapter we present a global set–oriented multi–level scheme which serves as the basis for most of the algorithms presented in this thesis. The primary variant of this scheme was proposed in [26]. It is based on a subdivision technique which allows the construction of a covering of the so–called *relative global attractor* of a given dynamical system $f : Q \subset \mathbb{R}^n \to \mathbb{R}^n$ in a given compact domain $Q \subset \mathbb{R}^n$ up to a prescribed accuracy. The relative global attractor is important for our considerations because it contains in particular all invariant sets of f within Q (a set $A \subset Q$ is called invariant with respect to f if $f(A) = A$). This fact will be utilized for the construction of the dynamical system for the underlying optimization problem: all dynamical systems which are used in combination with the subdivision algorithms have in common that the set of interest – e.g. the minimizer of an objective function in the context of scalar optimization – is contained in the invariant set.
In practice we would like to work with dynamical systems which are dependent on a parameter – e.g. the step size of the damped Newton method for the computation of zeros. This is the reason why we have to extend the basic algorithm to the situation where one is interested in the computation of the *common* invariant sets of a finite number of different dynamical systems $f_1, \ldots, f_s$. In fact, we generalize the analytical results in [26] to this situation and develop a subdivision algorithm for the outer approximation of the common global attractor of $f_1, \ldots, f_s$ inside a compact subset of state space (Section 2.2).
Moreover, we prove convergence of this abstract multi–level algorithm (Section 2.2), discuss its implementation (Section 2.3) and make some estimates of the computational effort (Section 2.4). The latter discussion will show that the subdivision techniques are restricted to moderate dimensions n of the parameter space of the dynamical system.
Parts of this chapter grew out of [106] and [30], for each of which the author has given a substantial contribution.

2.2 The Algorithm

We consider a finite collection of discrete dynamical systems of the type

$$x_{j+1} = f_\ell(x_j), \quad j = 0, 1, 2, \ldots,$$

where we assume for simplicity that each $f_\ell : \mathbb{R}^n \to \mathbb{R}^n$ $(\ell = 1, \ldots, s)$ is a diffeomorphism. The purpose is to develop a set oriented numerical method for the approximation of those subsets of state space which are invariant for the *entire collection* of dynamical systems. More precisely we want to approximate a subset $A \subset \mathbb{R}^n$ such that

$$f_\ell(A) = A \quad \text{for } \ell = 1, \ldots, s.$$

With this technique we generalize a known subdivision algorithm for the computation of invariant sets of single dynamical systems. See [26] or [27], [28] for generalizations of this approach. This new version of the subdivision algorithm will be e.g. the basis for the detection of zeros of a given function using Newton's method with different step sizes as the family of dynamical systems. In fact, in this case the set A consists of all the fixed points of the family of (damped) Newton iterations. See Chapter 3 for more information.

Relative Global Attractors

Given a compact subset $Q \subset \mathbb{R}^n$ the subdivision algorithm in [26] allows to compute the backward invariant set

$$A_{Q,f} = \bigcap_{j \geq 0} f^j(Q) \tag{2.2.1}$$

of a dynamical system $f : Q \to \mathbb{R}^n$. We now modify this definition to the present context in which we have to consider a finite set of different dynamical systems.

We begin by introducing some notations. Denote by

$$\Omega = \{1, 2, \ldots, s\}^{\mathbb{N}_0}$$

the set of all sequences of the symbols $\{1, 2, \ldots, s\}$. For $\omega = (\omega_i) \in \Omega$ set $\omega^j = (\omega_0, \omega_1, \ldots, \omega_{j-1})$ and define for $j \geq 1$

$$f_{\omega^j} = f_{\omega_{j-1}} \circ \cdots \circ f_{\omega_0}.$$

Definition 2.2.1 Let $f_1, \ldots, f_s : \mathbb{R}^n \to \mathbb{R}^n$ be diffeomorphisms and let $Q \subset \mathbb{R}^n$ be compact. Then we define the *relative global attractor* of $f_1, ..., f_s$ with respect to Q as

$$A_{Q,f_1,\ldots,f_s} = \bigcap_{\omega \in \Omega} \bigcap_{j \geq 1} f_{\omega^j}(Q) \cap Q. \tag{2.2.2}$$

Observe that $A_{Q,f_1,\ldots,f_s}$ is contained in the intersection of all the standard relative global attractors A_{Q,f_i}, see (2.2.1). Moreover, it contains every set A which is invariant for each f_ℓ, $\ell = 1, \ldots, s$.

EXAMPLE 2.2.2 Let $f_\ell : \mathbb{R}^n \to \mathbb{R}^n$, $\ell = 1, 2$, be defined as

$$f_1(x) = 3x \quad \text{and} \quad f_2(x) = \frac{1}{2}x,$$

and let $Q \subset \mathbb{R}^n$ be a compact convex subset containing the origin. Then the relative global attractors are given by

$$A_{Q,f_1} = Q \quad \text{and} \quad A_{Q,f_2} = \{0\}.$$

It follows that in this case the relative global attractor is given by $A_{Q,f_1,f_2} = 0$.

The following basic facts on relative global attractors can immediately be obtained from the definitions.

LEMMA 2.2.3 *Let $A_{Q,f_1,\dots,f_s}$ be the relative global attractor for the dynamical systems $f_1, \dots, f_s$ with repsect to Q. Then*

$$A_{Q,f_1,\dots,f_s} = \left\{ x \in Q : f_{\omega^j}^{-1}(x) \in Q \quad \textit{for all } \omega \in \Omega \textit{ and } j \geq 0 \right\}.$$

In particular, $A_{Q,f_1,\dots,f_s}$ is backward invariant for every f_ℓ, that is,

$$f_\ell^{-1}(A_{Q,f_1,\dots,f_s}) \subset A_{Q,f_1,\dots,f_s} \quad \textit{for all } \ell \in \{1, \dots, s\}. \tag{2.2.3}$$

Computation of Relative Global Attractors

We now present an algorithm for the numerical computation of the relative global attractor $A_{Q,f_1,\dots,f_s}$ where $f_1, \dots, f_s$ are diffeomorphisms. Using a multi–level subdivision scheme this method produces a sequence of sets $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \dots$ where each $\mathcal{B}_k$ consists of finitely many subsets of Q covering the relative global attractor $A_{Q,f_1,\dots,f_s}$. In the following we will call the elements B of $\mathcal{B}_k$ *boxes*. By our construction, the diameter

$$\operatorname{diam}(\mathcal{B}_k) = \max_{B \in \mathcal{B}_k} \operatorname{diam}(B)$$

converges to zero for $k \to \infty$.

In order to guarantee convergence we have to assume that each dynamical system is applied infinitely often in the subdivision procedure. To make this precise we choose a sequence $\{u_k\}_{k=1}^{\infty}$ with $u_k \in \{1, \dots, s\}$ which has the property

$$|\{k : u_k = \ell\}| = \infty \quad \text{for each } \ell = 1, \dots, s. \tag{2.2.4}$$

We now describe the multi–level subdivision procedure in detail.

The Subdivision Algorithm

Let $\mathcal{B}_0$ be an initial collection of finitely many subsets of the compact set Q such that $\cup_{B \in \mathcal{B}_0} B = Q$. Then $\mathcal{B}_k$ is inductively obtained from $\mathcal{B}_{k-1}$ in two steps:

(i) Subdivision Construct from $\mathcal{B}_{k-1}$ a new system $\hat{\mathcal{B}}_k$ of subsets such that

$$\bigcup_{B\in\hat{\mathcal{B}}_k} B = \bigcup_{B\in\mathcal{B}_{k-1}} B$$

and

$$\operatorname{diam}(\hat{\mathcal{B}}_k) = \theta_k \operatorname{diam}(\mathcal{B}_{k-1}),$$

where $0 < \theta_{min} \le \theta_k \le \theta_{max} < 1$.

(ii) Selection Define the new collection $\mathcal{B}_k$ by

$$\mathcal{B}_k = \left\{ B \in \hat{\mathcal{B}}_k : \text{there exists } \hat{B} \in \hat{\mathcal{B}}_k \text{ such that } f_{u_k}^{-1}(B) \cap \hat{B} \neq \emptyset \right\}.$$

(Here u_k is an element of the sequence defined above, see (2.2.4).)

Our aim is to show that this algorithm indeed converges to the relative global attractor as k tends to infinity.

Let Q_k be the union of subsets in $\mathcal{B}_k$,

$$Q_k = \bigcup_{B\in\mathcal{B}_k} B.$$

In particular we have $Q_0 = Q$. In analogy to the proof of the convergence result for the classical subdivision procedure ([26]) we divide the proof into three parts. In a first step we show that the relative global attractor is always covered by the sets Q_k.

LEMMA 2.2.4 *Let $A_{Q,f_1,\dots,f_s}$ be the relative global attractor of $f_1, \dots, f_s$ with respect to Q. Then*

$$A_{Q,f_1,\dots,f_s} \subset Q_k \quad \textit{for all } k \in \mathbb{N}.$$

Proof: By definition, see (2.2.2), we have that $A_{Q,f_1,\dots,f_s} \subset Q_0 = Q$. Now suppose that there is an $x \in A_{Q,f_1,\dots,f_s} \subset Q_{k-1}$ such that $x \notin Q_k$. Then there is a box $B \in \hat{\mathcal{B}}_k$ with $x \in B$, and B is removed from the collection in step k. In particular, $f_{u_k}^{-1}(B) \cap Q_{k-1} = \emptyset$ and therefore $f_{u_k}^{-1}(x) \notin Q_{k-1}$. But this contradicts the fact that $f_{u_k}^{-1}(A_{Q,f_1,\dots,f_s}) \subset A_{Q,f_1,\dots,f_s} \subset Q_{k-1}$, see Lemma 2.2.3. ∎

In the next step we show that a subset $A \subset Q$ is contained in the relative global attractor $A_{Q,f_1,\dots,f_s}$ if it is backward invariant for each f_ℓ, $\ell = 1, \dots, s$.

LEMMA 2.2.5 *Let $A \subset Q$ be a subset which is backward invariant for each $f_1, \dots, f_s$, that is,*

$$f_\ell^{-1}(A) \subset A \quad \textit{for } \ell = 1, \dots, s.$$

Then A is contained in the relative global attractor of $f_1, \dots, f_s$, that is,

$$A \subset A_{Q,f_1,\dots,f_s}.$$

Proof: By assumption we have

$$A \subset f_\ell(A) \quad \text{for all } \ell \in \{1, \ldots, s\},$$

and this implies that

$$A \subset f_{\omega^j}(A) \quad \text{for all } \omega \in \Omega \text{ and } j \geq 0.$$

Moreover $A \subset Q$ and therefore

$$A \subset \bigcap_{\omega \in \Omega} \bigcap_{j \geq 1} f_{\omega^j}(A) \cap Q \subset \bigcap_{\omega \in \Omega} \bigcap_{j \geq 1} f_{\omega^j}(Q) \cap Q = A_{Q, f_1, \ldots, f_s}.$$

□

Now the Q_k's form a nested sequence of compact subsets of Q and therefore the limit

$$Q_\infty = \bigcap_{k=0}^{\infty} Q_k$$

does exist. We now show that Q_∞ is backward invariant for each dynamical system f_ℓ ($\ell = 1, \ldots, s$).

LEMMA 2.2.6 *The set Q_∞ is contained in Q and backward invariant for each f_ℓ ($\ell = 1, \ldots, s$), that is,*

$$f_\ell^{-1}(Q_\infty) \subset Q_\infty \quad \textit{for } \ell = 1, \ldots, s.$$

Proof: Obviously $Q_\infty \subset Q$. For contradiction suppose that there is an $\ell \in \{1, ..., s\}$ and a $y \in Q_\infty$ such that $f_\ell^{-1}(y) \notin Q_\infty$. Since Q_∞ is compact it follows that there is a $\delta > 0$ with

$$d(f_\ell^{-1}(y), Q_\infty) > \delta.$$

Here d denotes the usual distance between a point and a set. Thus, there is an $N \in \mathbb{N}$ such that

$$d(f_\ell^{-1}(y), Q_k) > \delta/2 \quad \text{for all } k \geq N.$$

Now $y \in Q_\infty$ and therefore there exist boxes $B_k(y) \in \mathcal{B}_k$ with $y \in B_k(y)$ for all $k \in \mathbb{N}_0$. Since $\lim_{k \to \infty} \operatorname{diam}(\mathcal{B}_k) = 0$ and since f_ℓ is continuous there exists an $m > N$ such that $u_m = \ell$ and $f_\ell^{-1}(B_m(y)) \cap Q_m = \emptyset$. (Here we have used the property (2.2.4) of the sequence $\{u_k\}$.) By the selection step of the subdivision algorithm this is a contradiction to the fact that $y \in Q_\infty \subset Q_{m+1}$. □

Combining the lemmas we now show that the subdivision algorithm indeed converges to the relative global attractor $A_{Q, f_1, \ldots, f_s}$.

PROPOSITION 2.2.7 *Let $A_{Q, f_1, \ldots, f_s}$ be the relative global attractor of $f_1, ..., f_s$ with respect to $Q \subset \mathbb{R}^n$. Then the subdivision algorithm converges to $A_{Q, f_1, \ldots, f_s}$, that is,*

$$A_{Q, f_1, \ldots, f_s} = Q_\infty.$$

Proof: Lemma 2.2.4 states that $A_{Q,f_1,...,f_s} \subset Q_k$ for all $k \in \mathbb{N}$ which implies that $A_{Q,f_1,...,f_s} \subset Q_\infty$. By Lemma 2.2.6 Q_∞ is a backward invariant subset of Q for each $f_1, ..., f_s$. Therefore Lemma 2.2.5 implies that $Q_\infty \subset A_{Q,f_1,...,f_s}$ and we obtain

$$A_{Q,f_1,...,f_s} \subset Q_\infty \subset A_{Q,f_1,...,f_s}$$

as desired. □

Observe that the limit Q_∞ does not depend on the particular sequence $\omega = \{u_k\}$ which is chosen in the subdivision algorithm. The only criterion which has to be satisfied is (2.2.4). Moreover, the following example shows that in general

$$A_{Q,f_1,...,f_s} \neq \bigcap_{j\geq 1} f_{\omega^j}(Q) \cap Q,$$

where $\omega^j = (u_1, u_2, \ldots, u_j)$.

EXAMPLE 2.2.8 Let $Q = [-1, 1]$ and consider the dynamical systems $f_i : \mathbb{R} \to \mathbb{R}$ $(i = 1, 2)$ with $f_1(x) = 0.5x$ and $f_2(x) = 2x$. Then we choose the sequence

$$\{u_k\}_{k=1}^\infty = \{2, 2, 1, 2, 2, 1, 2, 2, 1, \ldots\}.$$

Obviously this sequence satisfies (2.2.4), but it is easy to see that

$$\bigcap_{j\geq 1} f_{\omega^j}(Q) \cap Q = [-1, 1] \neq \{0\} = A_{Q,f_1,f_2}.$$

REMARKS 2.2.9 (a) Since Q_∞ is the limit of the Q_k's, we can reformulate Proposition 2.2.7 as

$$\lim_{k\to\infty} h(A_{Q,f_1,...,f_s}, Q_k) = 0,$$

where $h(B, C)$ denotes the standard Hausdorff distance between two compact sets $B, C \subset \mathbb{R}^n$.

(b) It is obvious that the speed of convergence to the relative global attractor will in general crucially depend on the choice of the sequence $\{u_k\}$. In fact, for an efficient numerical realization one would expect that the u_k's should not be constant for a large number of successive indices k.

(c) Proposition 2.2.7 can be extended to the case where one has countably many different dynamical systems f_ℓ with $\ell \in \mathbb{N}$. However the sequence $\{u_k\}$ still has to be chosen in such a way that (2.2.4) holds for each $\ell \in \mathbb{N}$.

2.3 Realization

In this section we give a brief desciption of a possible implementation of the subdivision procedure. For details we refer to [26] and [66].

Realization of the Subdivision Step For the realization of the subdivision algorithm we use n-dimensional boxes. Every box $B \subset \mathbb{R}^n$ can be represented by a center $c \in \mathbb{R}^n$ and a radius $r \in \mathbb{R}^n_+$ such that

$$B = B(c, r) = \{x \in \mathbb{R}^n \,:\, |x_i - c_i| \leq r_i \; \forall i = 1, \ldots, n\}.$$

We start the subdivision algorithm typically with a single box $\mathcal{B}_0 = \{B\}$. In the k-th subdivision step we subdivide each box $B(c, r) \in \mathcal{B}_k$ of the current box collection in the simplest case by one bisection with respect to the j-th coordinate, where j is varied cyclically, that is, $j = ((k-1) \mod n) + 1$. This division leads to two boxes $R_-(c^-, \hat{r})$ and $R_+(c^+, \hat{r})$, where

$$\hat{r}_i = \begin{cases} r_i & \text{for } i \neq j \\ r_i/2 & \text{for } i = j \end{cases}, \quad c_i^{\pm} = \begin{cases} c_i & \text{for } i \neq j \\ c_i \pm r_i/2 & \text{for } i = j \end{cases}.$$

Of course it is also possible to make more bisections per iteration step. See next section for a detailed discussion.

The collections $\mathcal{B}_k$ can easily be stored in a binary tree. In Figure 2.1 a representation of three subdivision steps in three dimensions ($n = 3$) together with the corresponding sets $Q_k, k = 0, 1, 2, 3$, is shown. Note that each $\mathcal{B}_k$ is completely determined by the tree structure and the initial box $B(c, r)$. Using this scheme, the memory requirements grow only linearly in the dimension n of the problem.

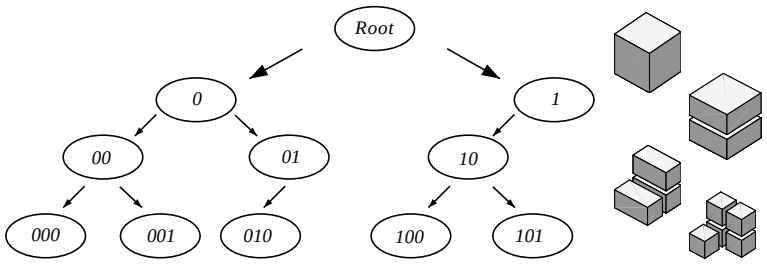


Figure 2.1: The data structure used for the subdivision techniques.

Realization of the Selection Step For the selection step, we have to decide whether or not the preimage of a given set $B_i \in \mathcal{B}_k$ has a nonzero intersection with another set $B_j \in \mathcal{B}_k$, i.e. whether or not

$$f^{-1}(B_i) \cap B_j = \emptyset \tag{2.3.5}$$

holds. Obviously, this can hardly be done exactly except for trivial mappings f. For more complex systems we have to use some kind of discretization. Motivated by similar approaches in the context of cell–mapping ([64]), we choose a finite set of *test points* in each box $B_j \in \mathcal{B}_k$ and replace condition (2.3.5) by

$$f(x) \notin B_i \text{ for all test points } x \in B_j. \tag{2.3.6}$$

Within each box, the test points are typically chosen by one of the following strategies:

(i) The test points lie on an a priori specified fixed grid within each single box. This typically works quite well in low dimensional problems.

(ii) The test points are chosen at random. This strategy is chosen in higher dimensional problems.

(iii) More recently an adaptive choice for the set of test points has been suggested in [66] where Lipschitz estimates on the underlying dynamical system are taken into account. This method works particularly well for dynamical systems which do not arise via a discretization of an underlying ordinary differential equation.

2.4 Estimates of the computational effort

Here we give some estimates of the computational effort of the subdivision algorithm for typical examples from real applications. The effort of the subdivision algorithm is determined by the total number of function evaluations, which in turn depends on the number of test points per box on the one hand and on the total number of boxes that are considered in the course of the computation on the other hand. For the latter, the following estimation is crucial:
Suppose we are given a function $f : \mathbb{R}^n \to \mathbb{R}^n$ and a set $\mathcal{M}$ – i.e. the object of interest given as the relative global attractor $A_{Q,f}$ of f and a compact set $Q \subset \mathbb{R}^n$ – of dimension[1] $1 \leq m \leq n$ [2]. Further let a box $B \in \mathcal{B}_l$, $l \in \mathbb{N}$, be given which contains a part of $\mathcal{M}$ and suppose the subdivision process is realized via bisection according to one coordinate in each iteration step. For sufficiently small B we can expect that appoximately 2^m of the 2^n subboxes of B intersect $\mathcal{M}$ (see Figure 2.2).

For sufficiently large l we can thus expect the number of boxes after n subdivision steps to be approximately

$$|\mathcal{B}_{l+n}| \approx 2^m |\mathcal{B}_l|.$$

Hence for one iteration step we can assume an *expansion factor* χ defined as

$$\chi \approx \sqrt[n]{2^m},$$

which fits quite well to the observation made by the author in numerous computations. Using these assumptions the expected number of boxes which intersect $\mathcal{M}$ in each iteration step is given by

$$\begin{aligned} |\mathcal{B}_0| &= 1 \\ |\mathcal{B}_l| = |\mathcal{B}_l(n,m)| &\approx \sqrt[n]{2^m} |\mathcal{B}_{l-1}| \approx (\sqrt[n]{2^m})^l, \quad l = 2, 3, \ldots, \end{aligned} \tag{2.4.7}$$

[1] The definition of the dimension of a set can be found e.g. in [38]. For simplicity the reader may think of a manifold.

[2] For $m = 0$, i.e. when $\mathcal{M}$ is a set of discrete points, the following estimates are too simple and do not coincide with observations made in practice.

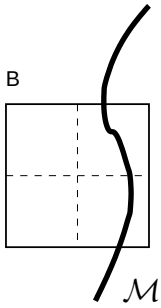


Figure 2.2: Example: two subboxes of B cover some part of $\mathcal{M}$ after two iteration steps.

Table 2.1: Approximate (round up) numbers of boxes $e(n, m, d)$ which have to be evaluated for several dimensions of the dynamical system f (n) and of the set $\mathcal{M}$ (m) for several iteration steps. Here one bisection per iteration step was used (i.e. $b = 1$).

depth d	f_1 $n = 2, m = 1$	f_2 $n = 10, m = 1$	f_3 $n = 5, m = 2$	f_4 $n = 10, m = 2$
$1 \cdot n$	5	28	19	41
$2 \cdot n$	15	84	94	202
$5 \cdot n$	150	864	6404	$1.3 \cdot 10^4$
$10 \cdot n$	4940	$2.8 \cdot 10^4$	$6.5 \cdot 10^6$	$1.4 \cdot 10^7$

in case the computation starts with one single box. Hence the expected number $e(n, m, d)$ of boxes which have to be evaluated up to iteration step d – i.e. the total number of boxes which intersect $\mathcal{M}$ in each iteration step – is given by

$$e(n, m, d) = \sum_{i=1}^{d} 2|\mathcal{B}_{i-1}|$$

Table 2.1 shows values of $e(n, m, d)$ for some typical (moderate) dimensions of the state space n of the dynamical system f and of the dimension m of the set $\mathcal{M}$. An example for a function f_1 with $n = 2$ and $m = 1$ is a multi–objective optimization problem of the form

$$\min_{x \in \mathbb{R}^2} f_1(x) = (g_1(x), g_2(x)),$$

where $g_1, g_2 : \mathbb{R}^2 \to \mathbb{R}$. See Chapter 6 for more information.

Furthermore, if the time for a function call can be approximated, then $e(n, m, d)$ can be used to estimate the total running time of the subdivision algorithm. For example, let the time for a function call be given by one second (0.001 seconds). Then

the computation of 20 iterations of the subdivision algorithm using a dynamical system $f_1 : \mathbb{R}^2 \to \mathbb{R}^2$ with $m = 1$ takes approximately 22 hours (80 seconds), if every box is evaluated by a 4×4-grid of test points. The computation of 25 subdivision steps using a dynamical system $f_3 : \mathbb{R}^5 \to \mathbb{R}^5$ with $m = 2$ will last approximately 17 hours (60 seconds), if for every box 10 randomly chosen points are taken. If 50 iteration steps have to be computed for the same model, the expected running time is 752 days (18 hours).

Since the number of bisections b which are done in each iteration step affects (slightly) the total number of boxes which have to be evaluated, we will now discuss the proper value of b in order to reduce the total running time.
To do this, we have to make some generalizations on the estimation $e(n, m, d)$. Note that if d_1 iteration steps are computed using b_1 bisections, the same box size is reached within $d_2 = d_1 \frac{b_1}{b_2}$ iteration steps using b_2 bisections per iteration – if the same initial box collection is used. The expected approximate number $e_b(n, m, d)$ of boxes which have to be evaluated up to iteration step d using b bisections is thus given by

$$e_b(n, m, d) = \sum_{i=1}^{d} 2^b |\mathcal{B}_{i-1}| = 2^b \chi^{(i-1)b}.$$

In the following we want to minimize $e_b(n, m, d)$. Suppose we are given a box collection $\mathcal{B}_l$ and a set $\mathcal{S}$ of possible number of bisections – say $\mathcal{S} = \{1, \ldots, n\}$. The minimal number of boxes which have approximately to be evaluated up to the succeeding d bisection steps (let e.g. d be the least common multiple of all elements of $\mathcal{S}$) is given by

$$\min_{b \in \mathcal{S}} \sum_{i=1}^{d_b} 2^b |\mathcal{B}_{l+(i-1)}|, \tag{2.4.8}$$

where $d_b = \frac{d}{b}$. Since $|\mathcal{B}_{l+(i-1)}| \approx \chi^{b(i-1)} |\mathcal{B}_l|$, problem (2.4.8) can be approximated by

$$\min_{b \in \mathcal{S}} 2^b |\mathcal{B}_l| \sum_{i=1}^{d_b} \chi^{b(i-1)}. \tag{2.4.9}$$

The computation of the minimum is simple since only few "candidates" for b have to be considered. See Table 2.2 for the optimal number of bisections for values of χ in $[1, 1.92]$.

Table 2.3 shows the expected number of boxes to be evaluated for dynamical systems of the same dimension as in Table 2.1 (for all examples the "optimal" number of bisections is $b = 2$). It turnes out that the difference gets smaller with smaller fraction $\frac{m}{n}$ – and hence with smaller expansion factor χ.
In practice the *observed* expansion factor

$$\chi_o^i = \frac{|\mathcal{B}_i|}{|\mathcal{B}_{i-1}|}.$$

Table 2.2: Optimal number b of bisection steps for different values of the expansion factor χ. Here it is assumed that the expansion is fixed for the next d steps.

Number of bisections	Expansion Factor
$\chi = 1$	$b = 1$
$\chi \in I_2 \approx [1, 1.61]$	$b = 2$
$\chi \in I_3 \approx [1.62, 1.83]$	$b = 3$
$\chi \in I_4 \approx [1.84, 1.92]$	$b = 4$

Table 2.3: Number of boxes $e_2(n, m, d)$, i.e. $b = 2$, which have approximately to be evaluated for several dimensions of the dynamical system f (n) and of the set $\mathcal{M}$ (m) for several iteration steps. Note that the difference to the values of $e_1(n, m, d)$ is hardly noticable when the fraction $\frac{m}{n}$ is small.

depth d	f_1 $n = 2, m = 1$	f_2 $n = 10, m = 1$	f_3 $n = 5, m = 2$	f_4 $n = 10, m = 2$
1	4	27	23	38
2	12	81	81	188
5	124	834	(not applicable)	$1.2 \cdot 10^4$
10	4092	$2.7 \cdot 10^4$	$5.6 \cdot 10^6$	$1.3 \cdot 10^7$

is typically higher than the "ideal" one. This is in particular the case when the dynamics of the system f inside the domain $\mathcal{B}_0$ is complicated or when the convergence of f towards $A_{Q,f}$ is slow (note that a box $B \in \mathcal{B}_l \subset \mathbb{R}^n$ has $3^n - 1$ neighbor boxes). Hence χ_o^i should be used. In computations the use of the "optimal" bisection number was observed to be advantageous in comparison to $b = 1$. Savings of up to 25% of the total running time have been noticed.

Conclusion The estimates made above lead to the conclusion that the use of the subdivsion techniques is fairly time consuming regarding the huge number of boxes which have to be evaluated until the granularity of the box collection is small enough. This seems to be the price one has to pay for the global nature of the method. In spite of this fact the set oriented approach permits the computation of "real world" optimization problems (for this we refer e.g. to the following sections). Nevertheless, these techniques should in general be viewed as a tool to obtain an "overview" of the global dynamics of the underlying system. A box collection computed by the subdivision algorithm can serve as a basis for other techniques as pointwise iteration methods or (set oriented) continuation. One probably convincing example is the problem of global zero finding within a compact set Q, which is described in detail

in Chapters 3 and 4: starting with $\mathcal{B}_0 \approx Q$ one can compute a box collection $\mathcal{B}_l$ which invariably covers the set of zeros within Q. When the boxes are small enough – or if it is clear that a box contains exactly one root – a classical iteration scheme like Newton's method can be used to locate this point. In this case the subdivision procedure gives the proper initial conditions for the solver. Another example is the class of recovering algorithms described in Chapter 6, which is developed to compute the connected component of the set of substationary points which contains a particular Pareto point (which was located e.g. via the use of DS–Subdivision).

Chapter 3

Location of Zeros A: $g : \mathbb{R}^n \to \mathbb{R}^n$

3.1 Introduction

In many applications in natural science or engineering the problem arises to detect all the zeros of a given smooth function $g : \mathbb{R}^n \to \mathbb{R}^n$ within a certain compact region $Q \subset \mathbb{R}^n$. One way to attack this problem is to choose randomly a large number of initial points inside Q and to apply a classical iteration scheme – e.g. a damped Newton method – to these points for a certain number of times. Using this approach there is a fair amount of uncertainty as to whether or not one has found all the zeros of g within Q. In principle this problem can be avoided by applying zero finding procedures which are based on *interval analysis*, see e.g. [1], [54] or [69]. However, due to the nature of these techniques there is a trade–off between rigor and computing time so that these methods are typically just applicable in the case where the dimension n is not too large.

Another way for locating all the zeros of a given function is given by *homotopy methods*, see e.g. [13], [2]. Although these methods are in general non–rigorous they can in certain cases be used to find the entire zero set, see e.g. [81], [119] in the situation where the underlying function is polynomial. Finally there are also global numerical techniques which are based on the adaptive refinement of Q into smaller subsets and by which all the zeros of g are approximated by outer coverings. See e.g. [121, 122, 31] or Chapter 4 in the case where $g : \mathbb{C} \to \mathbb{C}$ is a holomorphic function or [63] in the context of problems in global optimization.

In this chapter an approach is presented which in principle fits into the last category, that is, we propose an adaptive multi–level scheme for the outer approximation of the set of zeros of g inside a specified compact set Q. The approach was partly developed by the author and can also be found in [30]. An early version of these techniques is presented in [106]. The underlying idea is to view iteration schemes such as Newton's method as dynamical systems and then to apply the subdivision technique which is presented in Chapter 2. Observe that the zeros of g are fixed points for the iteration schemes and therefore represent particular low dimensional invariant sets.

In practice we like to work with iteration schemes with variable step size such as the damped Newton method $N_h(x)$, where $h \in \mathbb{R}^+$ is the damping factor, since this

makes the computations both more reliable and more efficient. Note that this also fits to the abstract framework of the subdivision algorithm which was proposed in the preceeding chapter: for every (fixed) damping factor h_i the roots of a given function g within Q are contained in the relative global attractor A_{Q,f_i} of the dynamical system $f_i(x) := N_{h_i}(x)$. Hence given s different step sizes, the aim is to compute the relative global attractor $A_{Q,f_1,\ldots,f_s}$ of all dynamical systems $f_1, \ldots, f_s$.

The theoretical results in Chapter 2 are not restricted to the situation where the underlying dynamical systems are produced by iteration schemes, and in Section 3.2 we show how these general results apply in the particular context of zero finding. Moreover we discuss the effect of choosing different step sizes on the behavior of the subdivision algorithm by a couple of motivating examples. This experience leads to the development of two algorithms based on different a priori step size strategies and of an adaptive algorithm based on the Armijo step size (see Section 3.2 for both). In principle these algorithms allow us to determine arbitrarily close coverings of the set of zeros of the function g. However, in practice we would switch to a classical iteration scheme as soon as a particular level of refinement has been reached. Finally we compare and discuss the numerical efficiency of these different approaches, and we also compare our method with a global zero finding procedure based on a routine taken from the NAG[1] library (Section 3.3).

3.2 The Algorithms

Based on the theoretical results that we have developed so far we now consider the problem of finding all the zeros of a given function $g : \mathbb{R}^n \to \mathbb{R}^n$ inside a specified compact subset $Q \subset \mathbb{R}^n$. We motivate our techniques with the following two examples.

EXAMPLES 3.2.1 (a) First we consider the function $g : \mathbb{R}^2 \to \mathbb{R}^2$

$$g(x_1, x_2) = \begin{pmatrix} 4x_1(x_1^2 + x_2 - 11) + 2(x_1 + x_2^2 - 7) \\ 2(x_1^2 + x_2 - 11) + 4x_2(x_1 + x_2^2 - 7) \end{pmatrix}$$

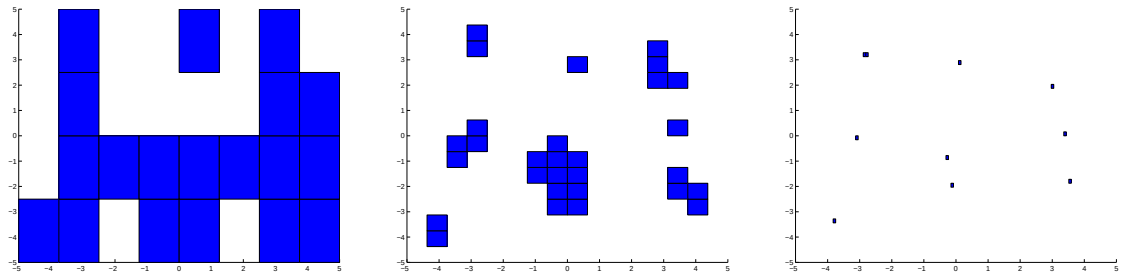
which is the gradient of an objective function proposed in [59]. We would like to find all the zeros of g within the compact set $Q = [-5, 5] \times [-5, 5]$.

In this example we set $s = 1$ and use the classical Newton function Ng as the only underlying dynamical system. The result of the computations is quite promising, see Figure 3.1. All the nine zeros within the given domain Q can be located after only a few subdivision steps (Figure 3.1(c)). In the computations we have chosen five test points in each box $B \in \mathcal{B}_k$ according to strategy (ii) in Section 2.3 for an evaluation of $Ng(B)$.

(b) As a second example we consider the following function $g : \mathbb{R}^2 \to \mathbb{R}^2$

$$g(x_1, x_2) = \begin{pmatrix} x_1^3 - 3x_1x_2^2 - x_1 + \frac{1}{\sqrt{2}} \\ -x_2^3 + 3x_1^2x_2 - x_2 \end{pmatrix}.$$

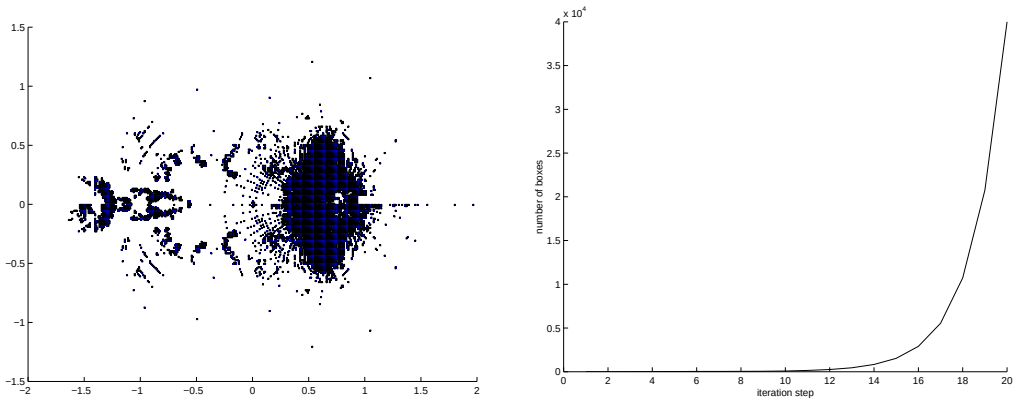
[1] `http://www.nag.com`



(a) $k = 5$ (b) $k = 8$ (c) $k = 11$

Figure 3.1: Application of the subdivision algorithm using Newton's method as the only dynamical system. The box collections $\mathcal{B}_5, \mathcal{B}_8$ and $\mathcal{B}_{11}$ are shown.

The aim is to find all the zeros inside $Q = [-5, 5] \times [-5, 5]$. Again we set $s = 1$ and use just the classical Newton method in the subdivision process. In the computations we have used nine test points per box lying on a 3×3 grid inside the box (i.e. strategy (i) in Section 2.3). The result of this computation after 20 subdivision steps is shown in Figure 3.2. It can be observed that



(a) Box covering (b) Numbers of boxes

Figure 3.2: (a) The box collection $\mathcal{B}_{20}$ consisting of 40648 boxes; (b) the number of boxes increases permanently with each subdivision step up to depth 20. Note that the box collection is not perfectly symmetric with respect to the x-axis. This is due to the occurrence of round off errors in the numerical computation of Ng.

the number of boxes is rapidly growing, and that no close covering of the set of zeros $\mathcal{Z} = \{(-1.251, 0), (0.625, 0.417), (0.625, -0.417)\}$ is obtained. The reason is that we do not just obtain a covering of $\mathcal{Z}$ alone but of all the invariant sets of Newton's method. For instance, in this case it can be shown that the Newton iteration has an asymptotically stable periodic orbit $\{(0, 0), (\frac{1}{\sqrt{2}}, 0)\}$. Thus, we have to develop step size strategies which allow us to eliminate the

computation of invariant sets of the iteration schemes besides their common fixed points.

Denote by

$$\mathcal{N}_{Q,g} = \{x \in Q : g(x) = 0\}$$

the set of all the zeros of g inside Q. If we are using the classical Newton method alone as the dynamical system for an approximation of $\mathcal{N}_{Q,g}$ then – as indicated by Example 3.2.1(b) – we may cover much more than just $\mathcal{N}_{Q,g}$ by the box collections obtained in the subdivision procedure. In order to avoid this problem we are going to use the damped Newton method

$$x_{j+1} = x_j - hDg(x_j)^{-1}g(x_j), \quad j = 0, 1, \ldots, \tag{3.2.1}$$

where $h > 0$ denotes the step length. More precisely, denoting the right hand side in (3.2.1) by

$$Ng_h(x) = x - hDg(x)^{-1}g(x),$$

we are going to work with finitely many different step lengths h_ℓ, $\ell = 1, \ldots, s$, and define s dynamical systems $f_\ell : \mathbb{R}^n \to \mathbb{R}^n$ $(\ell = 1, \ldots, s)$ by

$$f_\ell(x) = Ng_{h_\ell}(x).$$

By this construction we have

$$\mathcal{N}_{Q,g} \subset A_{Q,f_1,\ldots,f_s}, \tag{3.2.2}$$

where $A_{Q,f_1,\ldots,f_s}$ denotes the relative global attractor of $f_1, ..., f_s$ with respect to Q, see (2.2.2), and therefore we can compute an outer covering of the set of zeros of g by an application of the subdivision algorithm to $f_1, \ldots, f_s$.

Remark 3.2.2 Observe that formally the convergence result in Proposition 2.2.7 does not apply in this context since the mappings $f_1, \ldots, f_s$ will in general not be diffeomorphisms on the set Q. However, in practice we expect to rapidly lose neighborhoods of the singular points of $f_1, \ldots, f_s$ within the subdivision process. Moreover Proposition 2.2.7 certainly applies to the damped Newton method in the case when we choose Q to be a compact neighborhood of the set $\mathcal{N}_{Q,g}$ of regular zeros of g within Q.

The following result shows that for different step sizes h_1 and h_2 the dynamical systems f_1 and f_2 cannot have common periodic points of period greater or equal to two.

Proposition 3.2.3 *Let f_1 and f_2 be dynamical systems belonging to the step sizes h_1 and h_2. For $\bar{x} \in \mathbb{R}^n$ suppose that $g(\bar{x}) \neq 0$ and that the Jacobian $Dg(\bar{x})$ is invertible. Then*

$$f_1(\bar{x}) = f_2(\bar{x}) \Longrightarrow h_1 = h_2.$$

Proof: We compute

$$\begin{aligned} f_1(\bar{x}) &= f_2(\bar{x}) \\ \Longleftrightarrow \bar{x} - h_1 \underbrace{Dg(\bar{x})^{-1} g(\bar{x})}_{=:y \neq 0} &= \bar{x} - h_2 Dg(\bar{x})^{-1} g(\bar{x}) \\ \Longleftrightarrow (h_1 - h_2) y &= 0 \\ \Longrightarrow h_1 &= h_2. \end{aligned}$$

□

REMARK 3.2.4 Suppose that Q is chosen in such a way that $f_1, \ldots, f_s$ are diffeomorphisms on a neighborhood of Q. Then we expect that generically equality will hold in (3.2.2) if $s \geq 2$ and the different step sizes h_ℓ ($\ell = 1, \ldots, s$) are chosen randomly in $(0, 1]$. In fact, Proposition 3.2.3 indicates that common invariant sets of $f_1, \ldots, f_s$ do not contain periodic points which on the other hand would at least be generic for C^1-diffeomorphisms. (This is Pugh's celebrated Closing Lemma, see e.g. [112]). Also observe that the smaller we choose the step size h the closer is the map Nf_h to the identity.

The Algorithms A crucial aspect for an application of the subdivision algorithm lies in the selection procedure for the different step sizes h_ℓ. If the step size is not varied at all then we expect to obtain in the limit not just the zeros of g but also additional invariant sets of the (damped) Newton's method (see [34, 92] and also Example 3.2.1(b)). On the other hand we expect to lose this additional dynamical behavior when we use quite small step sizes (see [60, 70] and Remark 3.2.4). But in this case the number of boxes in the covering of $A_{Q,f_1,\ldots,f_s}$ will grow significantly and therefore one has to choose a step length control mechanism which balances these two different goals.

We now describe our first basic algorithm. In the following $g : \mathbb{R}^n \to \mathbb{R}^n$ is the function for which we would like to find all the zeros inside the compact set $Q \subset \mathbb{R}^n$.

Algorithm A In this algorithm we prescribe a priori two step lengths $h_1, h_2 \in (0, 1]$ together with four integers $q_1, q_2, n_1, n_2 \in \mathbb{N}$. Then we work with two dynamical systems

$$\begin{aligned} f_1 : \mathbb{R}^n \to \mathbb{R}^n, \quad f_1(x) &= (Ng_{h_1})^{q_1}(x), \\ f_2 : \mathbb{R}^n \to \mathbb{R}^n, \quad f_2(x) &= (Ng_{h_2})^{q_2}(x). \end{aligned}$$

Using the integers n_1 and n_2 we define the sequence

$$\{u_k\}_{k=1}^{\infty} = \{\underbrace{1, \ldots, 1}_{n_1}, \underbrace{2, \ldots, 2}_{n_2}, \underbrace{1, \ldots, 1}_{n_1}, 2, \ldots\}$$

By this definition it is clear that the condition (2.2.4) is satisfied and therefore, in principle, the convergence of the subdivision algorithm to A_{Q,f_1,f_2} is guaranteed.

In the realization of Algorithm A we typically choose $(h_1, q_1) = (1, 1)$ and $(h_2, q_2) = (0.1, 10)$. That is, we choose the classical Newton method and a strongly damped version.

Algorithm A has the disadvantage that the numbers n_1 and n_2 have to be defined in advance and that no adjustment is made in the course of the subdivision procedure. However, in the case that the number of boxes is growing too fast then it would be desirable to adjust the step size in Newton's method according to this observation. This leads to

Algorithm B Set $(h_1, q_1) = (1, 1)$ and specify an integer $N \in \mathbb{N}$. Moreover choose a (small) step size $h_2 \in (0, 1)$ and $q_2 \in \mathbb{N}$. Then proceed in the kth step of the algorithm according to the diagram in Figure 3.3.

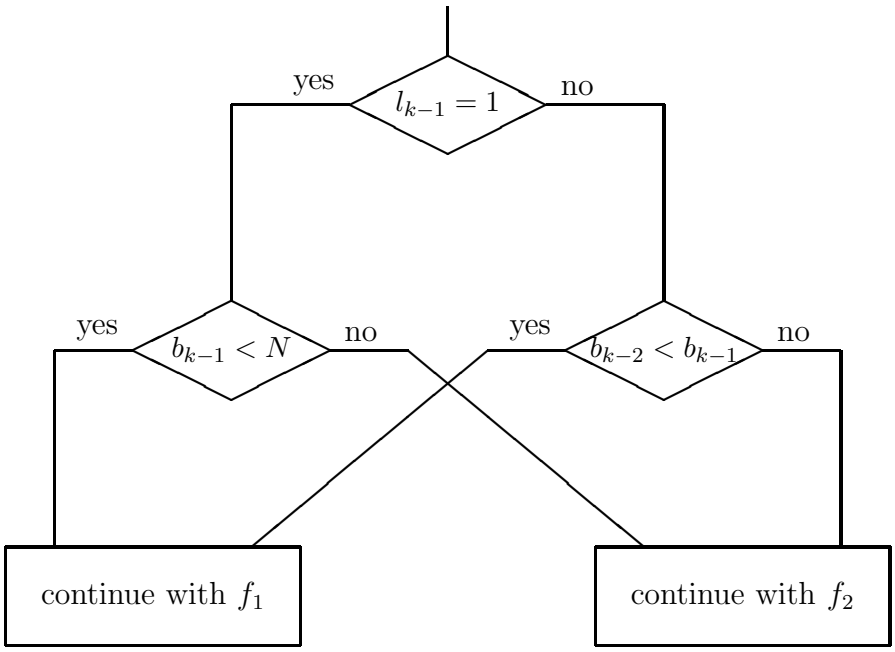
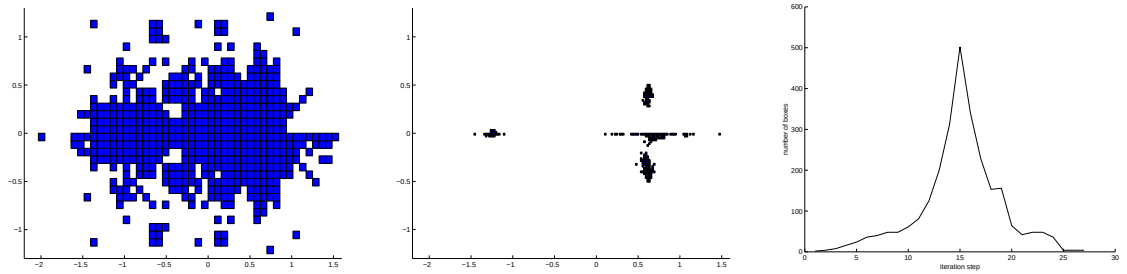


Figure 3.3: Schematic description of Algorithm B. b_k denotes the number of boxes in the box collection $\mathcal{B}_k$ and l_k the chosen step length.

During the subdivision procedure the number of boxes is monitored and h_2 is chosen as a damping parameter when the number of boxes b_{k-1} is increasing beyond the prescribed number N. Otherwise one proceeds with the classical Newton method, that is, with step length $h_1 = 1$. The reason for this strategy is that the occurence of a large number of boxes indicates that the subdivision procedure has found more than just the fixed points of Newton's method. A significant change in the damping parameter should lead to an elimination of these additional boxes. On the other hand, if the number of boxes is increasing again when the step length h_2 is repeatedly chosen then this is an indication for the fact that the contraction around the fixed points is not fast enough. In this case we switch back to the classical Newton method.

EXAMPLE 3.2.5 We now apply Algorithm B to Example 3.2.1(b) by choosing $N = 500$, $(h_1, q_1) = (1, 1)$ and $(h_2, q_2) = (0.1, 10)$. The results are shown in Figure 3.4. After 15 steps 502 boxes are computed by the subdivision algorithm. The algorithm then switches to the function $f_2 = (Ng_{0.1})^{10}$. In step 19 the number of boxes is again increasing so that the algorithm switches back to the classical Newton method. A close covering of the roots can be obtained after 23 subdivision steps.



(a) $k = 15$ (b) $k = 18$ (c) Number of boxes

Figure 3.4: Application of Algorithm B: the box collections $\mathcal{B}_{15}$, $\mathcal{B}_{18}$ and the number of boxes during the subdivision process are shown.

An Adaptive Algorithm We now present a refined version of Algorithm B. In this algorithm the step length h is chosen to be the Armijo step length (see [32]) and therefore the step size h becomes a function of x, i.e. $h = h(x)$. Moreover we specify in each subdivision step a power $q = q(x) \geq 1$ and apply the dynamical system $(Ng_{h(x)})^{q(x)}$ at the point x. Observe that this algorithm still fits into the underlying theoretical framework since in practice we only perform finitely many subdivision steps and since we are evaluating $(Ng_h)^q$ only at finitely many points.

We now state the underlying theoretical result which will allow us to find an appropriate choice for the power $q = q(x)$. Its proof can be found in [26].

PROPOSITION 3.2.6 *For $q \geq 1$ let A_{Q,f^q} be the global attractor of f^q relative to the compact set $Q \subset \mathbb{R}^n$ (cf. (2.2.1)). Moreover suppose that A_{Q,f^q} is an attracting compact hyperbolic set. Let $\rho \geq 1$ be a constant such that for each compact neighborhood $\tilde{Q}$ of A_{Q,f^q} we have*

$$h(A_{Q,f^q}, \tilde{Q}) \leq \delta \Longrightarrow \tilde{Q} \subset U_{\rho\delta}(A_{Q,f^q}).$$

Then the coverings Q_k obtained by the subdivision algorithm for f^q satisfy

$$h(A_{Q,f^q}, Q_k) \leq \operatorname{diam}(\mathcal{B}_k)(1 + \alpha + \alpha^2 + \cdots + \alpha^k). \tag{3.2.3}$$

Here $\alpha = C\rho\lambda^q/\theta_{min}$, where C is a constant, $\lambda \in (0, 1)$ is the number quantifying the contractivity of the hyperbolic set A_{Q,f^q} and θ_{min} is defined in the subdivision step of the subdivision algorithm.

REMARK 3.2.7 In Proposition 3.2.6 $h(E,F)$ denotes the Hausdorff distance between two compact sets E, F. Moreover $U_{\rho\delta}(A_{Q,f^q})$ is defined as

$$U_{\rho\delta}(A_{Q,f^q}) = \{y \in \mathbb{R}^n \quad : \quad \text{there is an } x \in A_{Q,f^q} \text{ such that } y \in W^s(x) \\ \text{and } \operatorname{dist}(x,y) < \rho\delta\}.$$

($W^s(x)$ is the *stable manifold* of the point x.)

Let us consider Proposition 3.2.6 in the specific context where A_{Q,f^q} is a fixed point of the damped Newton method Ng_h. In that case we have close to the fixed point

$$C \approx 1, \quad \rho \approx 1 \quad \text{and} \quad \lambda = 1 - h.$$

For a verification of this fact recall that fixed points of Ng_h are asymptotically stable with contraction rate $1 - h$.

Taking the estimate (3.2.3) into account we would like to choose q such that $\alpha < \epsilon$ for a specified (small) $\epsilon > 0$ which implies fast convergence of the algorithm. This leads to the following choice for the power q:

$$\epsilon > \alpha = \frac{\lambda^q}{\theta_{min}} \Longleftrightarrow \lambda^q < \epsilon\theta_{min} \Longleftrightarrow (1-h)^q < \epsilon\theta_{min} \Longleftrightarrow q > \frac{\ln(\epsilon\theta_{min})}{\ln(1-h)}.$$

This computation suggests that, as expected, one should choose a large q if the step length is small. Since in practice we do not want to exclude the case where $h = 1$, that is the case where we are working with the classical Newton method, we choose the power q as follows

$$q = \left\lceil \frac{\ln(\epsilon\theta_{min})}{\ln(\max(1-h), \delta)} \right\rceil, \tag{3.2.4}$$

where $\delta \in (0,1)$ is a prescribed constant.

Thus, we suggest the following adaptive subdivision algorithm:

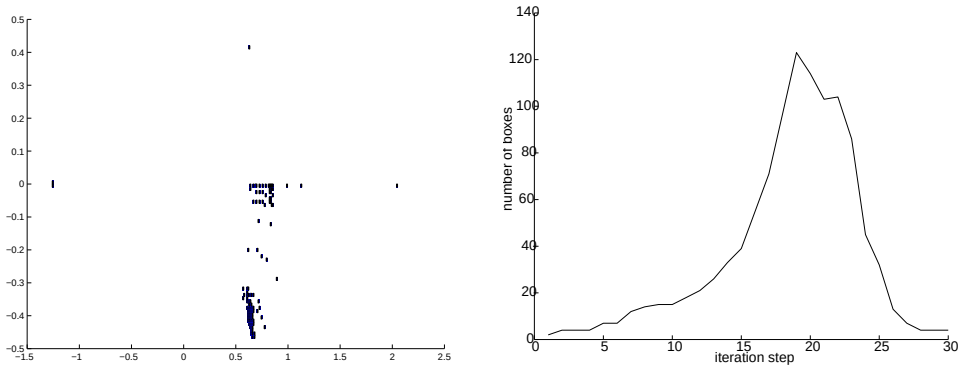
> in each subdivision step choose the Armijo step length $h = h(x)$ at the test point x. Then choose the power $q = q(x)$ according to (3.2.4) and apply $(Ng_{h(x)})^{q(x)}$ at the point x.

EXAMPLE 3.2.8 We reconsider Example 3.2.1(b) and show the result obtained by the adaptive algorithm in Figure 3.5. The maximum number of boxes calculated by this strategy is 123.

Finally we compare the three algorithms that we have described so far in Figure 3.6.

3.3 Numerical Results

We now investigate the computational efficiency of the adaptive algorithm by a comparison with another global root finding method based on the NAG routine `c05pbc()`. Our approach is expected to be particularly advantageous in the situation where the zeros of the given function g are not uniformly distributed inside the



(a) (b)

Figure 3.5: (a) The box collection $\mathcal{B}_{20}$ containing 114 boxes obtained by the adaptive algorithm. (b) Number of boxes in the box collections using the adaptive strategy.

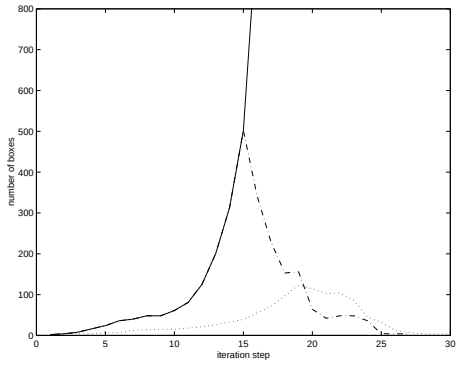


Figure 3.6: A comparison of the proposed algorithms: the respective number of boxes in the subdivision procedure is shown (classical Newton (solid), Algorithm B (dash-dotted) and the adaptive algorithm (dotted)).

compact set Q but rather occur in clusters. In fact, by the construction of the subdivision process these clusters should always be covered by the box collections. On the other hand it seems very unlikely to find all the zeros inside the clusters by an application of Newton's method to a certain number of randomly chosen initial conditions since the basins of attraction of the zeros are very different in size. We illustrate this fact by the Examples (a) and (b). Examples (c) and (d) are taken from the literature and indicate the efficiency of our apporach even in the case where the zeros do not occur in clusters.

In all the computations we have used the following set of parameter values for the adaptive algorithm:

$$\epsilon = 0.2 \quad \text{and} \quad \delta = 0.1,$$

see (3.2.4). Moreover the test points have been chosen randomly (i.e. strategy (ii) in Section 2.3).

(a) Consider the following test function:

$$g_1 : \mathbb{R}^2 \to \mathbb{R}^2$$

$$g_1(x,y) = \left(\psi(y) \prod_{i=1}^{40} (x - x_i), \quad \phi(x) \prod_{i=1}^{40} (y - y_i) \right),$$

where

$$x_i = y_i = \begin{cases} 0.1 + \frac{-10+i}{1000} & : \quad \text{for} \quad i = 1, ..., 20 \\ 0.9 + \frac{-30+i}{1000} & : \quad \text{for} \quad i = 21, ..., 40 \end{cases}$$

and

$$\psi(y) = \sin(4y), \quad \phi(x) = \sin(4x).$$

By construction g_1 possesses 1649 roots in the domain $Q = [-3,3] \times [-3,3]$ and most of them are concentrated in four clusters, Figure 3.7(c). In addition to an application of our adaptive algorithm we have tried to find all the roots by using the NAG–solver `c05pbc()` with 10.000 randomly distributed initial points. By this strategy only 300 roots were found (including some spurious zeros, see Figure 3.7(a)). In the right hand side of this figure the roots located by this method in one of the four clusters are shown. Observe that, as expected, almost all the zeros on the "boundary" of the cluster have been found but very few inside.

In Figure 3.7(b) we show the box collection obtained after 24 subdivision steps of the adaptive algorithm. All the roots are covered by the boxes. Moreover, by switching to the classical Newton method with a few initial conditions per box all the roots could be computed (see Figure 3.7(c)).

(b) In order to illustrate that our method also works in higher dimensions we now embed the previous example into an n-dimensional context: consider the following function

$$g_2 : \mathbb{R}^n \to \mathbb{R}^n,$$

$$g_2(x_1, ..., x_n) = (g_1(x_1, x_2), (x_3 - 3)^2, ..., (x_n - n)^2),$$

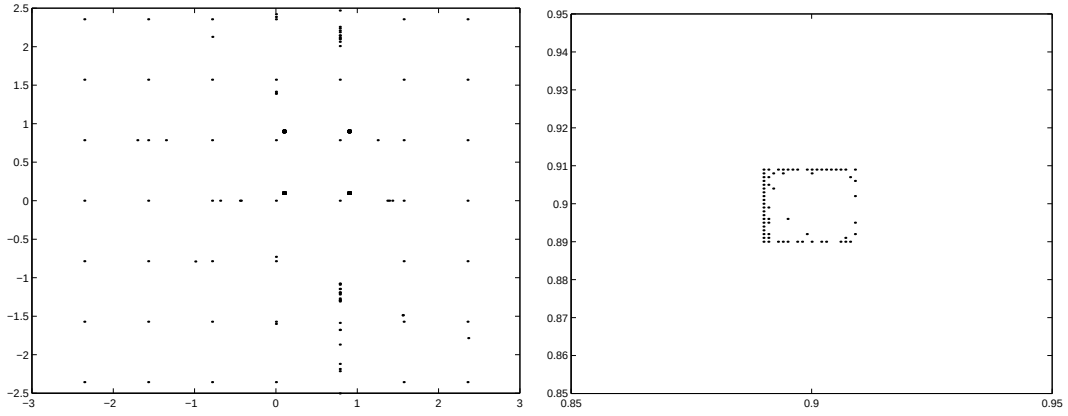
where g_1 denotes the function from the previous example. Using this function we have performed numerical tests up to dimension $n = 10$. Some of the results are presented in Table 3.1 and illustrate the efficiency of our approach.

(c) In this example we consider a polynomial function taken from [9]:

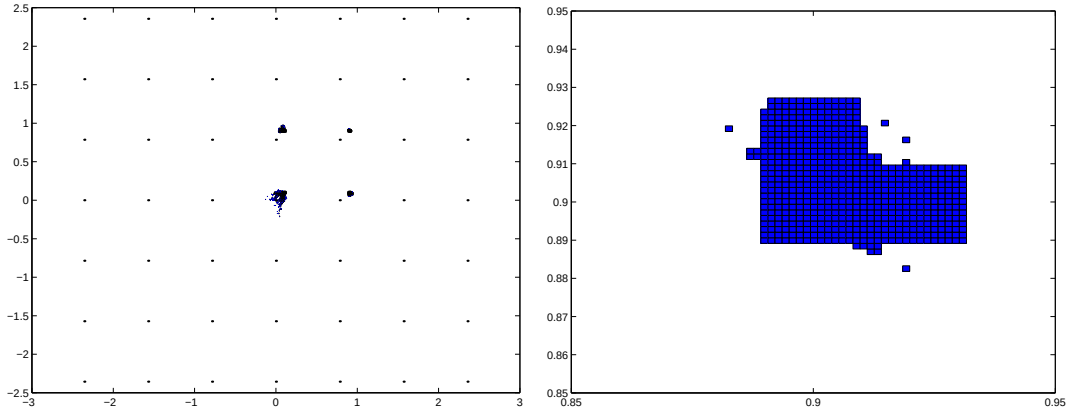
$$g_3 : \mathbb{R}^n \to \mathbb{R}^n,$$

$$g_3(x) = (f_1(x), .., f_n(x))$$

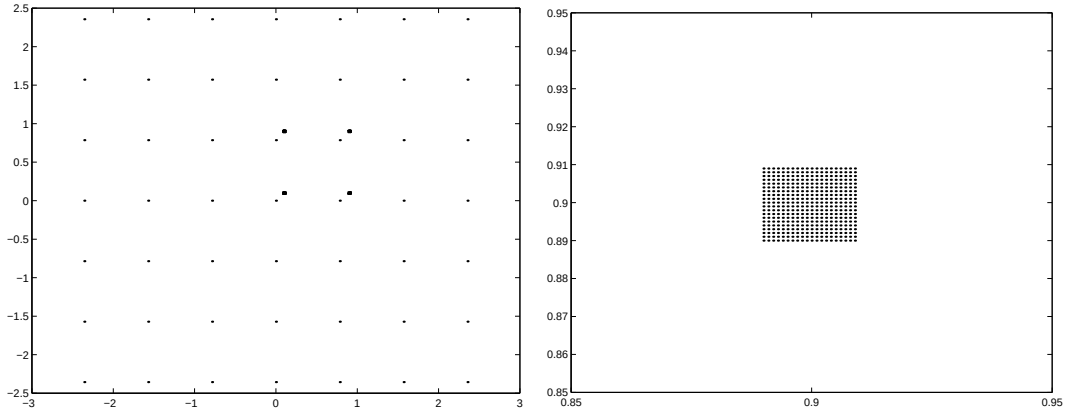
$$\text{where} \quad f_i(x) = -\sum_{i=1}^{n} x_i + \lambda x_i + x_i^2 - x_i^3.$$



(a) Zeros found by the NAG–solver `c05pbc()`



(b) The box collection $\mathcal{B}_{24}$ computed by the adaptive subdivision algorithm



(c) All the roots of g inside $Q = [-3, 3] \times [-3, 3]$

Figure 3.7: Numerical results for Example (a).

Table 3.1: Performance of the different zero finding procedures. In the table # IP denotes the number of test points per box (subdivision procedure) or the total number of initial points (NAG solver). # FC denotes the number of function calls and # DC the number of derivative calls. The computations have been done on a SUN Ultra 10 Workstation.

Dim	Method	# IP	# FC	# DC	CPU–time	# zeros found
5	Subdivision	5	$0.4 \cdot 10^7$	$2.6 \cdot 10^6$	5 min	1624
		12	$4.0 \cdot 10^7$	$27 \cdot 10^6$	47 min	1649
	NAG `c05pbc()`	50.000	$0.7 \cdot 10^7$	$0.8 \cdot 10^6$	5 min	570
		300.000	$4.0 \cdot 10^7$	$4.6 \cdot 10^6$	33 min	991
		400.000	$5.4 \cdot 10^7$	$6.2 \cdot 10^6$	44 min	1062
10	Subdivision	4	$0.6 \cdot 10^7$	$4.3 \cdot 10^6$	10 min	1609
		10	$9.6 \cdot 10^7$	$65 \cdot 10^6$	150 min	1649
	NAG `c05pbc()`	50.000	$0.7 \cdot 10^7$	$0.8 \cdot 10^6$	10 min	578
		600.000	$8.1 \cdot 10^7$	$9.2 \cdot 10^6$	124 min	1176
		800.000	$11 \cdot 10^7$	$12 \cdot 10^6$	164 min	1256

Here $\lambda \in [-50, 1000]$ is a bifurcation parameter and the equations can be viewed as a model describing speciation phenomena occuring in the evolution process. For $n = 8$ and $\lambda = 100$ the function has 6561 roots in $Q = [-40, 40]^n$. In Table 3.2 we have summarized our results in comparison with the NAG–solver. Since g_3 is polynomial, all the roots could in principle also be found by a global homotopy algorithm as implemented in the package PHC[2].

Finally we remark that we have also tried to find all the zeros for this example using the software GLOBSOL[3] taking the standard parameter values. However, this computation has not been successful since the maximal number of iteration steps has been exceeded in the course of the computation.

(d) Our final example is a test example taken from [80]:

$$g_4 : \mathbb{R}^n \to \mathbb{R}^n,$$
$$g_4(x) = (f_1(x), .., f_n(x)),$$
$$\text{with} \quad f_i(x) = n - \sum_{j=1}^{n} \cos(x_j) + i(1 - \cos(x_i)) - \sin(x_i).$$

For $n = 10$ there exist 10 roots inside the box $Q = [-0.3, 0.8]^n$. Since in this case the basin of attraction for Newton's method is quite large for every root we do not have a computational advantage in comparison with the NAG–solver. However, it can be seen that also in this case the computational effort is comparable, see Table 3.3.

[2]http://www.math.uic.edu/~jan/download.html
[3]http://interval.usl.edu/GLOBSOL

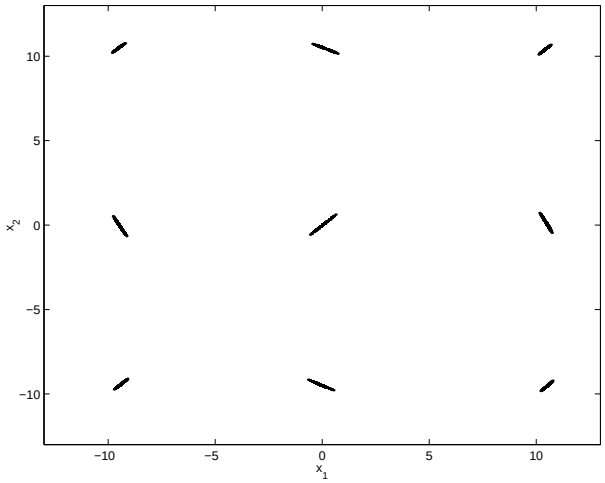


Figure 3.8: All the roots of g_3. A projection onto the first two coordinates is shown.

Table 3.2: Comparison for the test function g_3. The notation is the same as in Table 3.1.

Dim	Method	# IP	# FC	# DC	# zeros found
8	Subdivision	5	$2.2 \cdot 10^7$	$1.6 \cdot 10^7$	6463
		12	$5.9 \cdot 10^7$	$4.2 \cdot 10^7$	6561
	NAG `c05pbc()`	100.000	$2.8 \cdot 10^6$	$1.5 \cdot 10^5$	5716
		200.000	$5.5 \cdot 10^6$	$3.6 \cdot 10^5$	6128
		400.000	$1.1 \cdot 10^7$	$7.3 \cdot 10^5$	6392
		1.000.000	$2.7 \cdot 10^7$	$1.8 \cdot 10^6$	6522
		2.000.000	$5.6 \cdot 10^7$	$3.6 \cdot 10^6$	6556
		4.000.000	$8.3 \cdot 10^7$	$5.5 \cdot 10^6$	6560

Table 3.3: Comparison for the test function g_4. The notation is the same as in Table 3.1.

Dim	Method	# IP	# FC	# DC	# zeros found
10	Subdivision	3	9200	6800	8
		5	26747	19741	10
		8	137900	102700	10
		12	$1.0 \cdot 10^7$	$0.77 \cdot 10^7$	10
	NAG `c05pbc()`	250	13300	2300	6
		500	26400	4800	9
		750	39800	7400	10
		1000	52800	9800	10

3.4 An "Application": Scalar Optimization

In this section we propose a global approach for the numerical treatment of scalar optimization problems which uses the preceeding results for the localization of zeros as well as classical branching and bounding techniques. The presentation of this approach is placed in this chapter because it is the author's opinion that the results are worth more than just a remark but do (by far) not justify a separate chapter.
The algorithm which is stated below is designed for general unconstrained optimization problems and does not require particular properties of the underlying models. However, on account of this lack of "specialization" and due to the general restrictions of the subdivision techniques (see Section 2.4), the approach may have problems for higher dimensional models. For instance, it may be the case that the function has many local minima which cannot be discarded quickly by the selection process and hence the box collections will contain a large number of "false" boxes for a long time, resulting in slow convergence.
An alternative approach using subdivision techniques which allows the computation of certain "real world" optimization problems is presented in [111, 12].

To be more precise, we address in this section the following

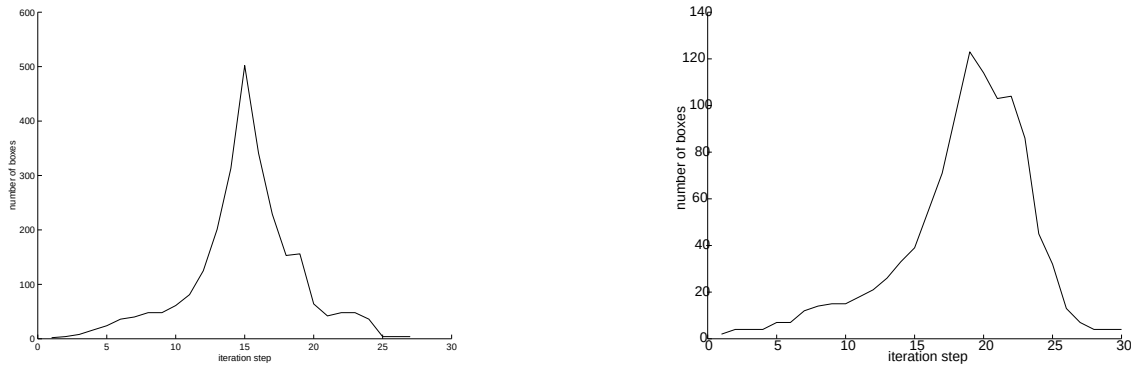
Problem Let a compact set $Q \subset \mathbb{R}^n$ and a continuous function $f : A \subset \mathbb{R}^n \to \mathbb{R}$, where $Q \subset A$, be given. The task is to find the set M_Q of global minimizers within Q, i.e.

$$M_Q := \{x^* \in \mathbb{R}^n \mid f(x^*) \leq f(y)\ \forall y \in Q\}. \tag{3.4.5}$$

One possible way to attack the problem is obviously to compute *all* the zeros of ∇f first and then to compare the function values of these candidates. This is in general not effective since f can contain many local minima. Another reason why it is not advising to compute all roots of ∇f in advance – if they are not required – by the use of the subdivision techniques is given by one characteristic of the approach: the subdivision algorithms which use Newton's method as dynamical system tend to generate many boxes in the "middle" part of the computation. Figure 3.4 shows two typical charts where the number of boxes $|\mathcal{B}_k|$ versus the iteration step k is plotted. In the first steps the collections typically contain few boxes. This indicates that the algorithm is able to compute a coarse localization of the region which contains all the roots of the underlying model quickly[4]. After several iteration steps the size of the boxes begins to increase rapidly. The reason for this seems to be that the covering is not tight enough such that Newton's method does not converge quadratically (recall that a box B has $3^n - 1$ neighbor boxes). This changes when the diameter of the boxes is small enough and hence the number of boxes is reduced rapidly within a few iteration steps.

[4]This is well known in the very restricted situation where g is a polynomial and $Q = [a, b]$ is large enough.

The typical problems which arise when using classical branching and bounding techniques are different: in the beginning of the computation it *can* be a difficult task to find tight bounds for big boxes or they are not helpfull in the sense that only few subsets can be deleted. Moreover, the process typically has to perform a lot of iteration steps until a "good" feasible point is detected to be in fact good since the accuracy of an approximate solution can only be measured by the upper and lower bounds of the boxes.



(a) Newton's Method $Ng_1(x)$

(b) adaptive strategy $Ng_h(x)^q$

Figure 3.9: Typical chart for the number of boxes vs. the iteration step: a peak appears in the "middle" part of the computation.

Basic Variant of the Algorithm

Motivated by the previous observations we now formulate a basic algorithm which combines the two techniques. Thus, it can be viewed as (another) variant of a branch&bound method.
Denote by `Newton-Step` an algorithm for the localization of zeros of the derivative of the objective function as described in Section 3.2, taking a box collection $\mathcal{B}_k$ as input and generating the succeeding collection $\mathcal{B}_{k+1}$ with

$$\operatorname{diam}(\hat{\mathcal{B}}_{k+1}) = \theta_{k+1} \operatorname{diam}(\mathcal{B}_k),$$

where $0 < \theta_{min} \leq \theta_{k+1} \leq \theta_{max} < 1$ (compare to Section 2.2). Let `Branch&Bound-Step` be defined analogously, e.g. by an algorithm described in [62]. Starting with a collection of finitely many boxes $\mathcal{B}_0$, the algorithm `DSBB` reads as follows:

Algorithm DSBB

Step 1:
$k = 0$
while $|\mathcal{B}_k| \leq C$
$\mathcal{B}_{k+1} =$ Newton-Step$(\mathcal{B}_k)$
$k = k + 1$
Step 2:
$sd = 0$
if $sd == 1$
$\mathcal{B}_{j+1} =$ Newton-Step$(\mathcal{B}_j)$
else
$\mathcal{B}_{j+1} =$ Branch&Bound-Step$(\mathcal{B}_j)$
if $|\mathcal{B}_{j+1}| > |\mathcal{B}_j|$
$sd = 1 - sd$

REMARKS 3.4.1 (a) In the examples we have computed we used the stopping criterion of the branch&bound-step for DSBB (or alternatively, the diameter of the boxes). Note that if the number of boxes remains permanently under the constant C, Step 2 will never be reached and hence no branch&bound technique will be applied. In this case this leads – when a prescribed maximal number of iterations is reached – to the comparison of the function values of the (few) points where the first order condition is fulfilled.

(b) In practice it was observed that the algorithm described above is more robust than the solitary use of the branch&bound techniques due to the (local) convergence of Newton's method. Herfore, it must be allowed to reintroduce boxes to the collection if an image $Ng_h(x)$ of a test point x is not contained in any box of the current collection.

(c) In [111] an algorithm which combines branch and bound methods with subdivision techniques for the computation of invariant sets of dynamical systems and its applicatility to global scalar optimization problems is presented. Since that algorithm is similar in spirit to the one described above, we just refer to the convergence result which is done in that work.

Numerical Results

Here we present the results for two (low dimensional) scalar optimization problems. For the realization of the branch&bound step we have used Lipschitz constants to estimate the bounds of every box. To estimate the Lipschitz constant L_B of f within a box $B \in \mathcal{B}_k$ we have done the following: starting from test points $TP(B) := \{x_1, \ldots, x_n\}$ we have estimated L_B as

$$L_B := \kappa \tilde{L},$$

where $\kappa > 1$ is a safety factor and

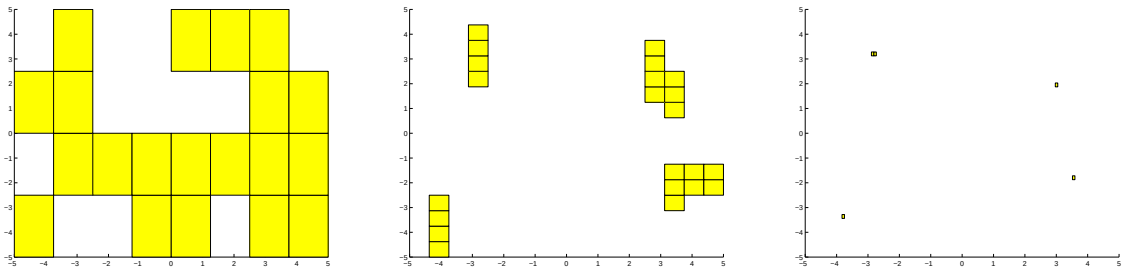
$$\tilde{L} := \max\left\{ \max_{i=1,..,n-1} \frac{|f(x_{i+1}) - f(x_i)|}{\|x_{i+1} - x_i\|_2}, \max_{i=1,..,n} \|\nabla f(x_i)\|_2 \right\}.$$

Example 1

First let us consider the following example taken from [59]:

$$f_1 : \mathbb{R}^2 \to \mathbb{R}$$
$$f_1(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

Figure 3.10 shows several box collections which are generated by an application of DSBB. Here, all four global minima within $Q = [-5, 5] \times [-5, 5]$ could be located. In Figure 3.11 the number of boxes which are generated in each iteration step by (a) using the dynamical system $Ng_h(x)^q$ for the computation of all the roots of ∇f – see Example (3.2.1)(a) – and (b) using the algorithm DSBB are compared indicating that the latter algorithm is able to compute the set M_Q much faster.



(a) 5 steps (b) 8 steps (c) 13 steps

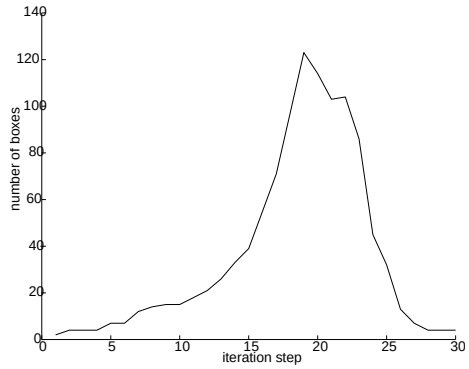
Figure 3.10: Coverings of the four minima of Example 1 for different iteration steps generated by algorithm DSBB. It can be observed that the roots ∇f which are not relevant in the optimization context get discarded soon in the course of the computation (compare to Figure 3.1).

Example 2

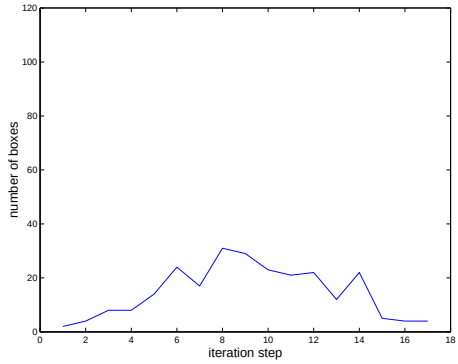
Next we we consider the following example (see [59]):

$$f_2 : \mathbb{R}^4 \to \mathbb{R}$$
$$f_2(x_1, x_2, x_3, x_4) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2$$
$$+ 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1),$$

By using DSBB the global minimizer $x^* = [1, 1, 1, 1]$ within the domain $Q = [-10, 10]^4$ could be found quickly.



(a) adaptive strategy $Ng_h(x)^q$



(b) use of `DSBB`

Figure 3.11: A comparison of the number of boxes which were produced in every iteration step by the different methods.

3.5 Conclusion

In this chapter we have presented a robust technique for the global zero finding of differentiable functions $g : \mathbb{R}^n \to \mathbb{R}^n$. This technique, including the use of different step sizes in the course of the computation in order to increase the performance, is based on the general framework of the subdivision algorithm which is described in Chapter 2. We have demonstrated the strength and the particular advantage of this technique by several examples. But it should be mentioned that, due to the global approach, these algorithms can only be applied for functions g with moderate dimension n. For higher dimensional problems a parallization strategy can be used to reduce the computational time. See [109] and [110] for more information.

Chapter 4

Location of Zeros B: $g : \mathbb{C} \to \mathbb{C}$

4.1 Introduction

Dynamical systems are frequently used as models for the temporal evolution of technical processes. In this case the stability analysis for an equilibrium state of the system naturally leads to the problem of finding all the zeros – or at least bounds on the set of zeros – of a certain analytic function $g : \mathbb{C} \to \mathbb{C}$. For instance, if there is no temporal delay to be taken into account in the underlying system then one has to consider the spectrum of the Jacobian of a vector field evaluated at the equilibrium. This can e.g. be done by using efficient eigenvalue solvers. However, if the technical process has to be modeled by a delay differential equation then the zero set of a non–polynomial but holomorphic function has to be approximated.

Motivated by this application we propose in this chapter – following [31] – a robust numerical method for the computation of all the zeros of a holomorphic function $g : \mathbb{C} \to \mathbb{C}$ within a certain bounded domain. The underlying idea is to use numerically the *argument principle* and to combine this properly with the subdivision algorithm to obtain an adaptive multi–level strategy. This way we construct tight box coverings of the set of zeros of g. The reliability of the method is due to the fact that by the argument principle one has to check whether or not a certain number is a nonvanishing integer, and this test is typically quite robust.

The underlying idea of this approach, namely to use the argument principle in combination with a subdivision procedure, already occurs in [121, 122]. However, we go one step further and propose an adaptive subdivision technique. Moreover, in contrast to [121, 122] we also describe in detail a specific numerical realization. This approach is similar in spirit to the topological methods developed in [91] and [120] where degree theory is the underlying concept for the approximation of the set of zeros of a given function. In fact, the argument principle as used here has to be viewed as the specific holomorphic situation for the computation of the degree of a mapping (cf. [24]). Further it should be mentioned that almost in parallel – and without the knowledge of the authors of [31] – the software package ZEAL (see [73]) was developed which basically includes the same functionality which is presented here.

A detailed outline of the chapter is as follows. In Section 6.2 we briefly restate the

argument principle. Then, in Section 4.3, we develop our adaptive multi–level algorithm, prove its convergence and discuss a particular numerical realization. Finally we present three examples indicating the robustness and efficiency of this numerical approach (Section 4.4).

4.2 Theoretical Background

Our numerical approach is based on an elementary fact from complex analysis namely the so–called "argument principle". For a given meromorphic function g this result states that the number of zeros minus the number of singularities of g (counting multiplicities) within a specified region in $\mathbb{C}$ can in principle be computed by a certain integration:

THEOREM 4.2.1 (ARGUMENT PRINCIPLE) *Let $g : U \to \mathbb{C}$ be a meromorphic nonconstant function on the open subset $U \subset \mathbb{C}$ and let γ be a closed curve on the boundary of a compact set K inside U. Finally denote by q_j, $j = 1, \ldots, n$, and by p_ℓ, $\ell = 1, \ldots, m$, the zeros resp. the singular points of g inside K. Then*

$$\frac{1}{2\pi i} \int_\gamma \frac{g'(z)}{g(z)} dz = \sum_{j=1}^{n} n(\gamma, q_j)\mu(q_j) - \sum_{\ell=1}^{m} n(\gamma, p_\ell)\mu(p_\ell),$$

where $n(\gamma, q_j)$, $n(\gamma, p_\ell)$ denote the winding numbers of the curve γ with respect to q_j resp. p_ℓ and $\mu(q_j)$, $\mu(p_\ell)$ are the multiplicities of q_j resp. p_ℓ.

In our applications we are particularly interested in the situation where g is a holomorphic function and where the winding number of γ is one. For this specific case we immediately obtain

COROLLARY 4.2.2 *Let $g : U \to \mathbb{C}$ be a holomorphic nonconstant function on the open subset $U \subset \mathbb{C}$ and let γ be a closed curve on the boundary of a compact set K inside U with winding number one for all the points surrounded by γ. Finally denote by q_j, $j = 1, \ldots, n$, the zeros of g inside K. Then*

$$\int_\gamma \frac{g'(z)}{g(z)} dz = 2\pi i \sum_{k=1}^{n} \mu(q_j) \tag{4.2.1}$$

where $\mu(q_j)$ are the multiplicities of the zeros q_j.

In the following we will denote by $\mu(g, \gamma)$ the right hand side in (4.2.1).

4.3 The Algorithm

Our aim is to find all the zeros of an analytic function g in one complex variable inside a given compact domain. The underlying idea of the algorithm is as follows. We start with a (big) rectangle B in $\mathbb{C}$ inside which we would like to find all the

zeros of g. Then we compute the integral in the left hand side in (4.2.1) where γ is a parametrization of the boundary of the rectangle B with winding number one. If this integral is zero then there are no zeros inside B and we are done. Otherwise we subdivide B into smaller rectangles and compute the integrals in (4.2.1) over their boundaries. Proceeding this way and disregarding rectangles for which the integral is zero we obtain a close covering of all the zeros of g inside B.

4.3.1 Description of a Basic Subdivision Scheme

We now formalize the idea stated above and present the abstract multi–level subdivision procedure in detail. The general structure of the following abstract algorithmic scheme is essentially known and can be found in the literature (see [121, 122]).

Basic Subdivision Scheme

Let $\mathcal{B}_0$ be an initial collection of finitely many rectangles in $\mathbb{C}$. Then $\mathcal{B}_k$ is inductively obtained from $\mathcal{B}_{k-1}$ in two steps:

(i) Subdivision step Construct from $\mathcal{B}_{k-1}$ a new system $\hat{\mathcal{B}}_k$ of rectangles such that

$$\bigcup_{B\in\hat{\mathcal{B}}_k} B = \bigcup_{B\in\mathcal{B}_{k-1}} B$$

and

$$\operatorname{diam}(\hat{\mathcal{B}}_k) = \theta_k \operatorname{diam}(\mathcal{B}_{k-1}), \tag{4.3.2}$$

where $0 < \theta_{min} \le \theta_k \le \theta_{max} < 1$. (Here $\operatorname{diam}(\mathcal{B}_k)$ denotes the diameter of the largest rectangle inside $\mathcal{B}_k$.)

(ii) Selection step Define the new collection $\mathcal{B}_k$ by

$$\mathcal{B}_k = \left\{ B \in \hat{\mathcal{B}}_k : \mu(g, \gamma_B) \neq 0 \right\}.$$

REMARKS 4.3.1 (a) If at some stage in the subdivision process a zero of g is lying on the boundary of one of the rectangles B then $\mu(g, \gamma_B) = \infty$. We explicitly allow this possibility in the selection step and keep the corresponding rectangle in this case.

(b) Observe that we do not require that the rectangles inside the collection $\mathcal{B}_k$ are disjoint. However, in our modification and realization of the basic subdivision scheme we are always working with collections having this property in order to avoid unnecessary numerical computations.

Denote by $z(g, B)$ the set of zeros of the holomorphic function g inside the rectangle B and let

$$Z_k = \bigcup_{B\in\mathcal{B}_k} B.$$

Observe that $\lim_{k\to\infty} Z_k$ does exist since the Z_k form a nested sequence of compact sets. Moreover, it is immediate from the construction of the general algorithm that the following result holds.

PROPOSITION 4.3.2 *An application of the basic subdivision scheme to the rectangle $B = \mathcal{B}_0$ yields a sequence of collections $\mathcal{B}_k$ such that*

$$\lim_{k\to\infty} h(Z_k, z(g, B)) = 0, \tag{4.3.3}$$

where $h(\cdot,\cdot)$ denotes the standard Hausdorff distance.

4.3.2 Adaptive Version of the Basic Subdivision Scheme: the QZ-40 Algorithm

We now present a modification of the basic subdivision scheme which will lead to a robust and efficient numerical realization. The crucial difference consists essentially of two ingredients: first an adaptive subdivision strategy is used and second we introduce an additional search step. The structure of the following algorithm is significantly different to that of the basic subdivision scheme. Therefore – in order to avoid any potential confusion – we will from now on use the letter R instead of B in the notation for rectangles or their collections, respectively.

The QZ-40 Algorithm

Let $\mathcal{R}_0$ be an initial collection of finitely many rectangles in $\mathbb{C}$. Then $\mathcal{R}_k$ is inductively obtained from $\mathcal{R}_{k-1}$ in three steps:

(i) Selection step For every rectangle $R \in \mathcal{R}_{k-1}$ denote by γ_R a parametrization of the boundary of the rectangle and compute the winding number $\mu(g, \gamma_R)$. Remove all rectangles R from $\mathcal{R}_{k-1}$ for which $\mu(g, \gamma_R) = 0$.

(ii) Search step Search for a zero inside each rectangle $R \in \mathcal{R}_{k-1}$ with $\mu(g, \gamma_R) = 2\pi i$ using Newton's method with a starting point inside R. If a zero is found then store this point and remove the rectangle R from the collection $\mathcal{R}_{k-1}$.

(iii) Adaptive subdivision step Construct from $\mathcal{R}_{k-1}$ a new system $\hat{\mathcal{R}}_k$ of rectangles according to a certain specified subdivision strategy satisfying (4.3.2). Then additionally subdivide each rectangle inside $\hat{\mathcal{R}}_k$ which is a subset of a rectangle $R \in \mathcal{R}_{k-1}$ with the property that $\mu(g, \gamma_R)/2\pi i > 2$. Let $\mathcal{R}_k$ be the resulting collection of boxes.

As in the case of the basic subdivision scheme the QZ-40 algorithm produces a nested sequence of compact sets Z_k covering the (remaining) zeros of g. Since the diameter of the boxes is shrinking to zero we have the following result:

PROPOSITION 4.3.3 *Suppose that all the zeros of g inside the rectangle R are simple. Then the QZ-40 algorithm applied to the rectangle $R = \mathcal{R}_0$ terminates after finitely many steps returning a complete list of all the zeros of g inside R.*

REMARKS 4.3.4 (a) It is an easy task to modify the algorithm in such a way that also zeros of higher multiplicity can be found.

(b) One has to expect that in general in the first subdivision steps the integers $\mu(g, \gamma_R)/2\pi i$ will be greater than two for almost every rectangle within the selected box collections. From this point of view it would seem natural to modify the subdivision step (iii) in such a way that additional subdivisions are just considered from a certain subdivision level $k > 1$ on. We have performed several numerical experiments along these lines, and these indicated that – somewhat counter–intuitively – the most efficient strategy is indeed to consider additional subdivisions right from the beginning.

Numerical Realization

A crucial part in the numerical realization of the QZ-40 algorithm is the computation of the integral (4.2.1) for the different rectangles. In our current implementation the computation of $\mu(g, \gamma_R)$ is done via an adaptive Romberg quadrature as described in [33] along the four different edges of the rectangle. More precisely we perform the three steps of the QZ-40 algorithm as follows:

- In a first step we parametrize each of the edges R_j of a rectangle by γ_j ($j = 1, 2, 3, 4$) and compute the value of

$$I_j \approx \Re \left(\frac{1}{2\pi i} \int_{\gamma_j} \frac{g'(z)}{g(z)} dz \right) \quad (j = 1, 2, 3, 4)$$

  via an adaptive Romberg quadrature, see Figure 4.1. Then

$$\mu(g, \gamma_R) \approx 2\pi i \sum_{j=1}^{4} I_j.$$

  Observe that the value of the tolerance in the adaptive Romberg scheme can be chosen quite large since we only have to decide whether or not the approximation of $\sum_{j=1}^{4} I_j$ is zero or a nonvanishing integer.

- In the search step we perform a search for a zero of g using the classical Newton method. This method either terminates with a zero inside the rectangle under consideration or the iteration is stopped if an iterate is more than a specified distance away from the center of the rectangle. In the examples which we considered it turned out to be quite efficient to choose at random a few (five, say) starting points for Newton's method inside the rectangle.

- The subdivision rule used in the adaptive subdivision step is simply given by bisection alternating between the two different coordinate directions. That is, if a rectangle is created via bisection with respect to the z_1-direction then, if necessary, it will be bisected in the next step with respect to the z_2-direction and vice versa. Obviously by this rule the condition (4.3.2) is satisfied.

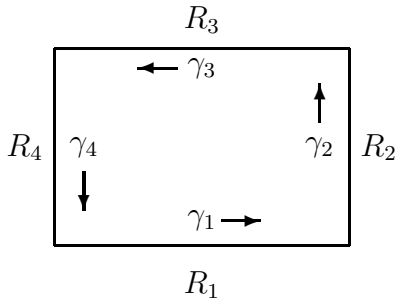


Figure 4.1: Boundary of the rectangle R and its parametrization $\gamma_R = \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4$.

4.4 Numerical Results

The following examples indicate both the efficiency and in particular the robustness of the QZ-40 algorithm.

4.4.1 Academic Example

We consider the function

$$g_1 : \mathbb{C} \to \mathbb{C}, \qquad g_1(z) = z^{50} + z^{12} - 5\sin(20z)\cos(12z) - 1,$$

and compute its zeros inside the rectangle $[-20.3, 20.7] \times [-20.3, 20.7]$. A couple of coverings of the zeros of g_1 produced by the QZ-40 algorithm are shown in Figure 6.7. The computations indicate that there are precisely 424 zeros of g inside the initial rectangle.

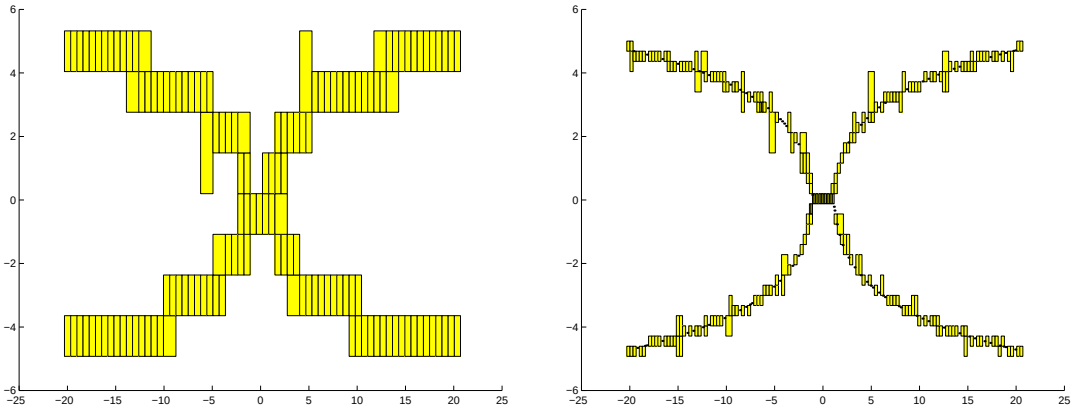
We have compared the numerical effort of the QZ-40 algorithm with the one obtained by two alternative approaches namely

- the classical Newton method performing at most 100 iterations with a certain number of initial points chosen at random. The numbers in Table 4.1 are averages over 20 different computations.
- the zero finding procedure `c05pbc` of the NAG library with a certain number of initial points chosen at random. In these computations we have specified a tolerance of 1e-15.

It can be observed, see Table 4.1, that the QZ-40 algorithm is much more efficient than these two approaches.

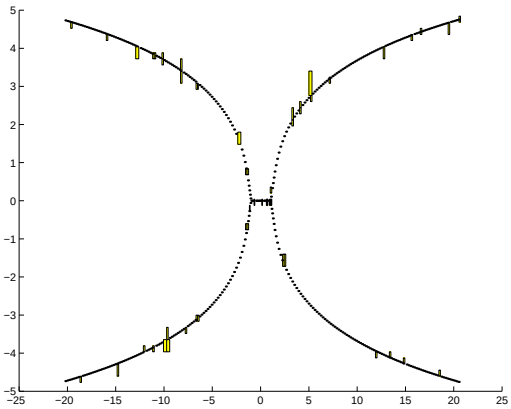
4.4.2 Stability of a Ring Oscillator

In [123] the detection of Hopf bifurcations in a ring consisting of coupled oscillators has been investigated. Taking the symmetry of the problem into account – i.e. by

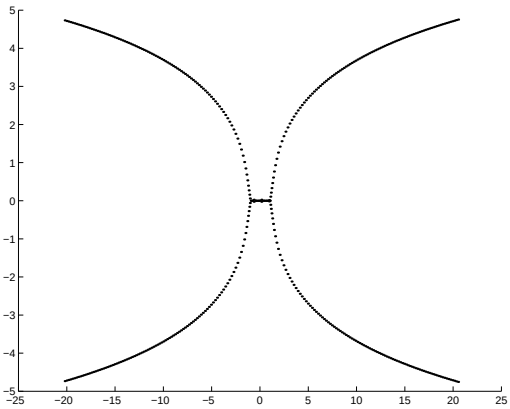


(a) 6 subdivisions

(b) 8 subdivisions



(c) 10 subdivisions



(d) 18 subdivisions

Figure 4.2: Box coverings of the set of zeros of g_1 obtained by the QZ-40 algorithm.

Table 4.1: Performance of the different zero finding procedures.

Method	# initial points	# function calls	# derivative calls	# zeros
Classical Newton	1500	140.130	140.130	153
	5.000	470.038	470.038	322
	7.000	655.879	655.879	361
	10.000	938.136	938.136	403
	15.000	1.405.903	1.405.903	419
NAG `c05pbc`	1500	454.856	6.966	177
	5.000	735.214	23.672	292
	7.000	1.033.345	32.209	336
	10.000	1.481.284	47.653	378
	15.000	2.238.868	71.276	406
QZ-40	–	89.619	89.619	424

restricting to the bifurcation of so–called *discrete rotating waves* – one has to find all the zeros of the following function:

$$g_2 : \mathbb{C} \to \mathbb{C},$$

$$g_2(\lambda) = \det \begin{pmatrix} -0.0166689 - 2.12 \cdot 10^{-14}\lambda & \frac{1}{60} + 6.0 \cdot 10^{-16} e^{-\tau\lambda}\lambda \\ 0.0166659 + e^{-\tau\lambda}(-0.000037485 + 6.0 \cdot 10^{-16}\lambda) & -0.0166667 - 6.0 \cdot 10^{-16}\lambda \end{pmatrix},$$

where we have chosen the delay to be $\tau = 2.0$. The result is shown in Figure 4.3. For more detailed information concerning this problem we refer to [123].

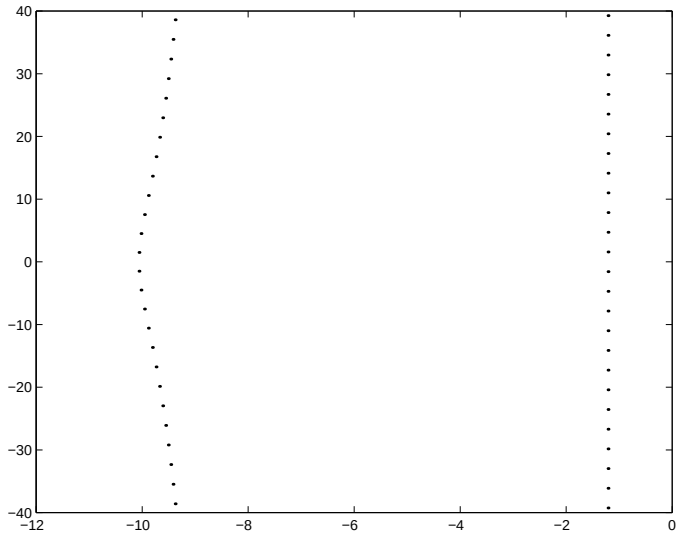


Figure 4.3: All the roots of g_2 within the rectangle $R = [-12, 0] \times [-40, 40]$.

4.4.3 Stability of an Annular Combustion Chamber

Within a project at the Corporate Technology Department of Siemens[1] (Munich), the stability of a flow inside an annular combustion chamber had to be analyzed. This lead to a model which can be written in the following compact form

$$\dot{x}(t) = \mathcal{A}x(t) + \mathcal{B}x(t-\tau), \tag{4.4.4}$$

where $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{n\times n}$ and $\tau \in \mathbb{R}_0^+$. The so–called *characteristic function* Δ of this system is given by

$$\Delta(\lambda) = \det(\lambda I - \mathcal{A} - \mathcal{B}e^{-\tau\lambda}). \tag{4.4.5}$$

It is well known (see e.g. [37]) that the trivial solution of (4.4.4) is asymptotically stable if

$$\sup\{Re(\lambda) : \Delta(\lambda) = 0\} < 0.$$

Thus, if a reliable stability analysis has to be performed, the structure of the set of zeros of the function Δ has to be investigated.

In a first simplified model this lead to the problem of finding all the zeros of the following holomorphic function:

$$\begin{aligned} \Delta_1 &: \mathbb{C} \to \mathbb{C} \\ \Delta_1(z) &= z^2 + Az + Be^{-Tz} + C, \end{aligned}$$

where A, B, C and T are real parameters. Motivated by the actual underlying model for the combustion chamber we choose the values

$$A = -0.19435, \quad B = 1000.41, \quad C = 522463 \quad \text{and} \quad T = 0.005.$$

In Figure 4.4 all the roots of Δ_1 found by the QZ-40 algorithm inside the rectangle $R = [-15000, 5000] \times [-15000, 15000]$ are shown.

More sophisticated considerations of the system lead to models of the form (4.4.4) with dimensions up to $n = 128$. When all roots within a prescribed region of interest are located, the next question arises: can the delay τ be adjusted in order to stabilize the system and, if possible, how can this be done. Figure 4.5 shows all the roots of a particular system (the (rescaled) underlying characteristic function is denoted by Δ_2) within $R = [-200, 600] \times [0, 3000]$ varying $\tau \in [0, 0.006]$. Note that standard pathfollowing techniques are not suitable in this context since it can occur that a root of the characteristic function enters the region of interest when varying τ, as shown in Figure 4.4 for Δ_2 ($Im(\lambda) \approx 1200$).

Figure 4.6 shows the movement of two zeros varying τ. The quasi–cyclic motion is due to the periodicity of the exponential function. For more information concerning the detection of the stability regions of parameter–dependent delay differential equations we refer to the next chapter.

[1]The author would like to thank the department, in particular Dr. Utz Wever and Dr. Qinghua Zheng, for the cooperation.

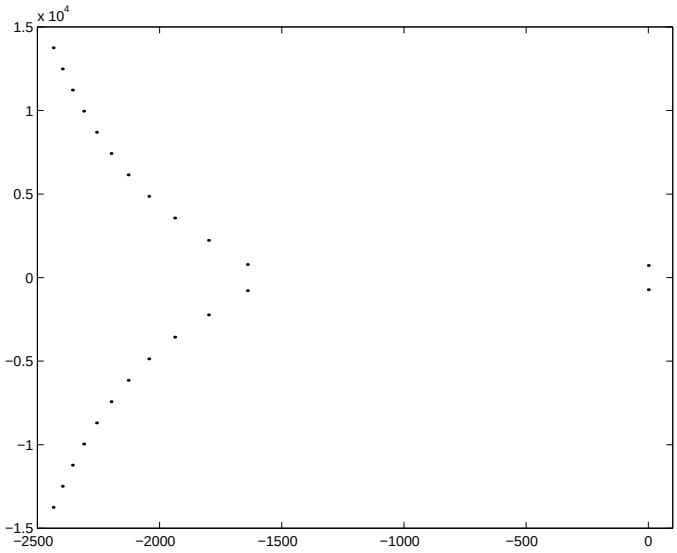


Figure 4.4: All the roots of Δ_1 inside $R = [-15000, 5000] \times [-15000, 15000]$.

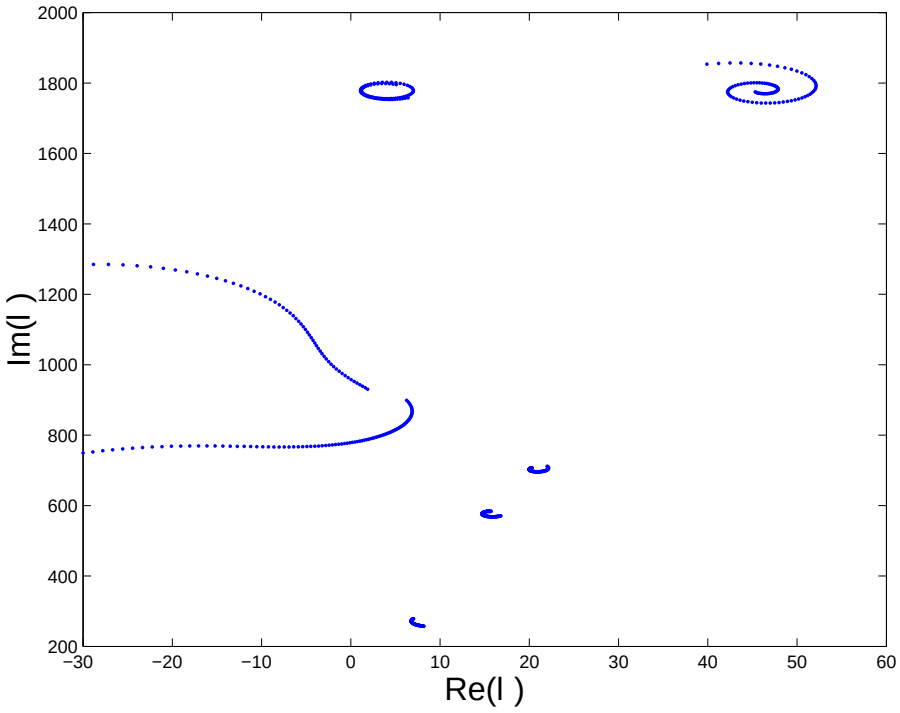


Figure 4.5: All the roots of Δ_2 inside $R = [-200, 600] \times [0, 3000]$ varying $\tau \in [0, 0.006]$.

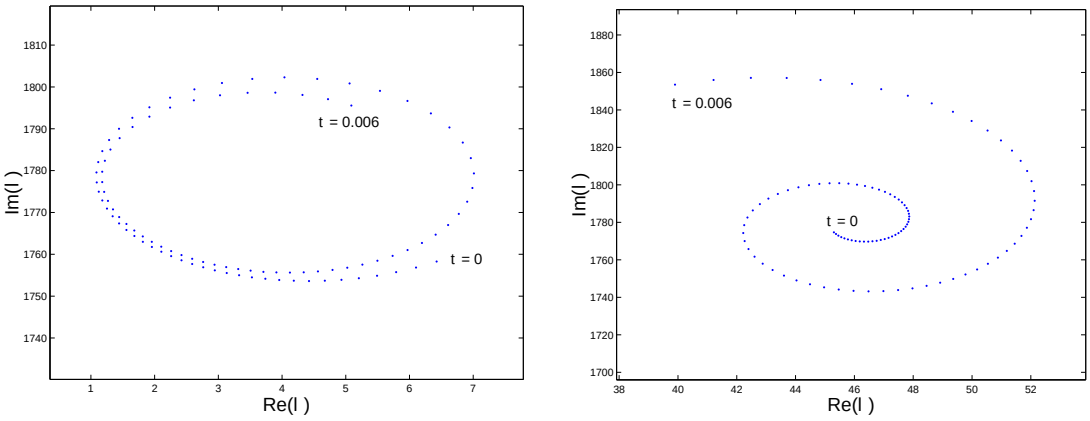


Figure 4.6: A zoom into Figure 4.5: localization of zeros varying $\tau \in [0, 0.006]$.

4.5 Conclusion

In this chapter we have presented a robust method for the localization of all zeros of a holomorphic function within a prescribed rectangle $R \subset \mathbb{C}$ which is based on a combination of the argument principle and the subdivision algorithm. Further, we have demonstrated its applicability to an academic example and two models which arise in applications. In the latter case the problem became apparent that one needs to locate the stability region of parameter–dependent delay differential equations. This will be the topic of the next chapter.

Chapter 5

Computing the Stability Regions of Delay Differential Equations

5.1 Introduction

In engineering and biological sciences, often systems with delays are studied. In most cases, the first step to understand the effect of the delay is to study the linear stability of the system. One way to do this is to analyse the characteristic function (see Section 5.2) of the system. Having a robust criterion for the stability of a particular adjustment of the system, the next question is to locate *all* possible stable settings of the system, which is the topic of this chapter.

Let us assume we are given a system of delay differential equations (DDEs) of the following form:

$$\frac{dx}{dt} = F(x(t), x(t-\tau_1), \ldots, x(t-\tau_m), \lambda), \tag{5.1.1}$$

where $\lambda \in \mathbb{R}^n$ is a vector of parameters. The task is to locate the region $\mathcal{S} \subset \mathbb{R}^n$ of parameters for which the trivial solution $x = 0$ is asymptotically stable.

As an introductory example let us consider balancing a stick (for details see [55] or [113]). Figure 5.1 shows the underlying mechanical model, which is the simplest possible model describing the "man–machine" system when somebody places the end of a stick on his fingertip and tries to move the lowest point of the stick in a way that its upper position should be stable.

Let the actual control force F_C be given in the form

$$F_C(t) = A\dot{x}(t-\tau) + Bx(t-\tau), \tag{5.1.2}$$

where τ is the delay of the human reflexes. Mathematically speaking, the task here is to choose the parameters A and B properly, that is, to determine A and B such that the upper position of the stick ($x = 0$) is stable. This is the case when all roots of the underlying characteristic function have a negative real part. We will come back to this example in Section 5.4.3, where we also show the proper values of A and B for all delays $\tau \in \mathbb{R}^+$.

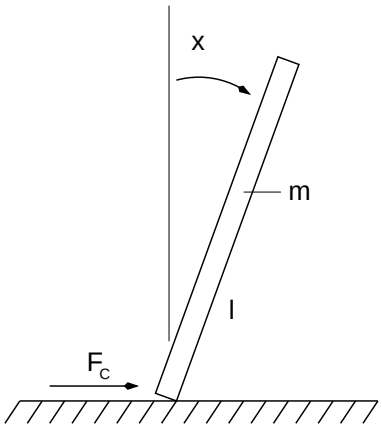


Figure 5.1: A mechanical model of the stick balancing problem.

In the literature there can be found a huge variety of algorithms for the computation of the stability regions of DDEs. One way to investigate the stability is to use Lyapunov's direct method (see references [10], [53] or [72]). In addition, there exist many methods which are based on the stability analysis of the characteristic function. The most common approaches to locate the stability domain are based on the works of Nyquist ([88]), Pontryagin ([93]) and Neimark ([86]). A good survey on these and further methods is given in [37] or [113]. Common to these different methods is that they are very effective on a particular class of DDEs but none of them can be used generally.

The numerical algorithm which is presented here fits in principle to the second branch. In particular we utilize the stability criterion which uses the characteristic function. To be more precise, we present a scheme which attacks the more general problem of the approximation of arbitrary subsets $\mathcal{S} \subset Q$ within a compact set $Q \subset \mathbb{R}^n$. This can be done since the stability regions can be formulated explicitly as a definite set. The resulting method uses subdivision techniques and is thereby global and very robust. Furthermore, since no particular structure of the DDE (5.1.1) is taken into account, the algorithm allows for the computation of the stability regions of a broad class of problems. On the other hand, it cannot compete with any of the algorithms mentioned above when applied to DDEs from their particular domains.

5.2 Theoretical Background

For sake of completeness we briefly state in this section the notations and facts which are required for the algorithm. For details we refer e.g. to [37] or [35].

The examples which are considered in this chapter are linear differential equations with constant coefficients and constant delays which can be stated in its general

form as follows:

$$\sum_{p=0}^{n}\sum_{j=0}^{m}\lambda_{pj}x^{(p)}(t-\tau_j)=0, \tag{5.2.3}$$

where λ_{pj} and τ_j are constants with $\tau_m > \tau_{m-1} > .. > \tau_1 > \tau_0 = 0$.

The characteristic function of (5.2.3) is given by

$$\Delta_\lambda(z)=\sum_{p=0}^{n}\sum_{j=0}^{m}\lambda_{pj}z^p e^{-\tau_j z} \tag{5.2.4}$$

The trivial solution $x = 0$ of (5.2.3) is asymptotically stable if

$$Z_\lambda := \{z\in\mathbb{C} \,:\, \Delta_\lambda(z)=0 \text{ and } Re(z)\geq 0\}=\emptyset. \tag{5.2.5}$$

Hence the stability region of DDE (5.2.3) can be stated explicitly as the following set:

$$\mathcal{S}_D=\left\{\lambda\in\mathbb{R}^{(n+1)\times(m+1)} \,:\, Z_\lambda=\emptyset\right\}. \tag{5.2.6}$$

5.3 The Algorithm

In this section we present an algorithm for the computation of the stability regions of parameter dependent DDEs. As described in the previous section, this problem can be formulated by the the more general problem of computing an (arbitrary) subset of a given domain. In the following we will present an adaptive scheme for the approximation of sets $\mathcal{S}\subset Q$, where $Q\subset\mathbb{R}^n$ is compact. We start with the formulation of an abstract subdivision scheme and will afterwards address its numerical realization.

Basic Subdivision Scheme

Let $\mathcal{B}_0$ be an initial collection of finitely many boxes. Further let all boxes be marked for subdivision. Then $\mathcal{B}_k$ is inductively obtained from $\mathcal{B}_{k-1}$ in two steps:

(1) Subdivision step

(a) subdivide all marked boxes $B\in\mathcal{B}_{k-1}$ and denote the resulting box collection by $\hat{\mathcal{B}}_k$.

(b) mark all boxes $B\in\hat{\mathcal{B}}_k$ that were subdivided in Step (1a).

(c) $\mathcal{B}_k := \{B\in\mathcal{B}_{k-1} \,:\, B \text{ is not marked}\}$.

(2) Selection step

for all marked boxes $B\in\hat{\mathcal{B}}_k$:

(a) unmark B.

(b) if $\exists x_1\in B\cap\mathcal{S}$:

– $\mathcal{B}_k := \mathcal{B}_k \cup B$.
– if further $\exists x_2 \in B : x_2 \notin \mathcal{S}$: mark B.

Denote by $\mathcal{S}_k$ the union of all boxes of $\mathcal{B}_k$, i.e.

$$\mathcal{S}_k = \bigcup_{B \in \mathcal{B}_k} B,$$

and let $\mathcal{S}_Q := \mathcal{S} \cap Q$. For simplicity of further statements let Q be a box. Note that $\lim_{k\to\infty} \mathcal{S}_k$ does exist in the Hausdorff–sense since the $\mathcal{S}_k$ form a nested sequence of compact sets. The following result is obvious by the construction of the algorithm.

PROPOSITION 5.3.1 *An application of the basic subdivision scheme to the set $\mathcal{B}_0 = Q$ yields a sequence of collections $\mathcal{B}_k$ such that*

$$\lim_{k\to\infty} h(\mathcal{S}_k, \mathcal{S}_Q) = 0,$$

where $h(\cdot,\cdot)$ denotes the standard Hausdorff distance.

In the course of the computation, the box collections $\mathcal{B}_k$ get refined on the boundary of the set of interest $\mathcal{S}$. Since $\mathcal{S}$ can in general *not* be described via a collection of boxes, it is crucial to have an effective strategy to mark the boxes which contain a part of $\mathcal{S}$ but are not contained in that set. In practice, the following implementation of the selection step turned out to be most effective.
Here, $TP(B)$ denotes a set of test points (see Section 2.3) within a box B with $|TP(B)| \geq 2$.

Selection step (alternate version)
for all marked boxes $B \in \hat{\mathcal{B}}_k$:

(a) unmark B.
(b) if $\exists x_1 \in TP(B) \cap \mathcal{S}$:
– $\mathcal{B}_k := \mathcal{B}_k \cup B$.
– if further $\exists x_2 \in TP(B) : x_2 \notin \mathcal{S}$: mark B.

else

mark all boxes $B_n \in \mathcal{B}_k : B_n \cap B \neq \emptyset$
mark all boxes $\hat{B}_n \in \hat{\mathcal{B}}_k : \hat{B}_n \cap B \neq \emptyset$

Before we continue we have to make some

REMARKS 5.3.2 (a) The set of test points $TP(B)$ for the evaluation of a box B can in general be chosen as described in Section 2.3. However, it was observed that typically one needs to take more test points than e.g. in the root finding context since the set of interest $\mathcal{S}$ is not described as an attractor of a dynamical system. Further, for the same reason, the efficiency of the "discretized" algorithm gets poor when the Lebesque measure of $\mathcal{S}$ is small

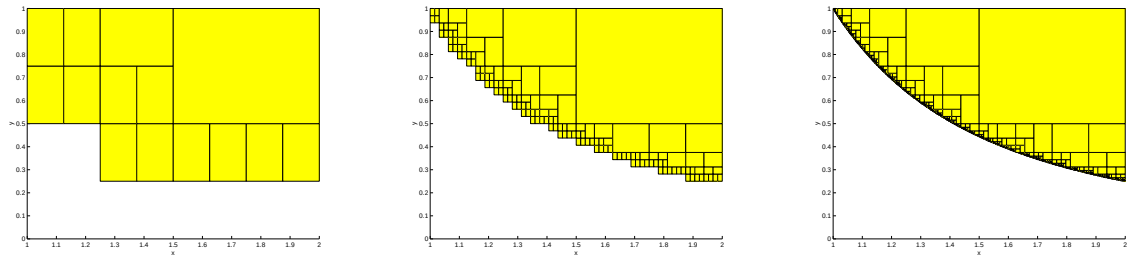
compared to the Lebesque measure of Q. In that case – and if the Lebesque measure of $\mathcal{S}$ is positive – a set-oriented continuation-like method (as described in a different context in Chapter 6) seems to be promising. However, this was not done yet and has to be tested in future.

(b) In the context of the localization of the stability regions of parameter dependent DDEs the set of interest is given by (5.2.6). A practical test which checks if a particular value of λ is a member of this set can be performed by computing all the zeros of the characteristic function Δ_λ within a suitable problem specific rectangle $Q \subset \mathbb{C}$, e.g. by the use of the algorithm $QZ40$ which is described in the previous chapter. Hence, no particular structure of the underlying DDE is required.

To make the approach more transparent let us consider a trivial example. Assume we want to detect the set $\mathcal{S}_1$ which is defined as follows

$$\mathcal{S}_1 := \left\{ x \in [1,2] \times [0,1] \ : \ x_2 \geq \frac{1}{x_1^2} \right\}. \tag{5.3.7}$$

Figure 5.2 shows appoximations of $\mathcal{S}_1$ for three different iteration steps starting with $\mathcal{B}_0 = [1,2] \times [0,1]$. It can be seen that boxes which are completely contained in $\mathcal{S}_1$ remain untouched in the course of the computation from a certain iteration depth. Moreover, it can be observed that the covering is beeing refined adaptively along the boundary of $\mathcal{S}_1$ (in the inside of $\mathcal{B}_0$).



(a) 4 steps (b) 10 steps (c) 18 steps

Figure 5.2: Adaptive refinement of the set $\mathcal{S}_1$ which is defined in (5.3.7).

5.4 Numerical Results

In this section we show approximations of the stability regions which are computed by the algorithm described above for three examples.

5.4.1 Example A

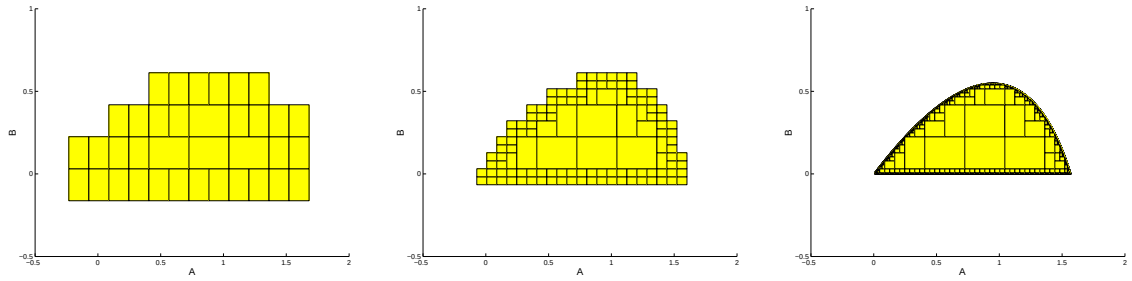
First we consider the following DDE (see [37]):

$$\ddot{x}(t) + A\dot{x}(t-1) + Bx(t-1) = 0, \tag{5.4.8}$$

where $\lambda = (A, B) \in \mathbb{R}^2$ has to be chosen such that the trivial solution is asymptotically stable. The characteristic function of (5.4.8) is given by

$$\Delta_1(z) = z^2 + Aze^{-z} + Be^{-z}$$

Figure 5.3 shows box collections for different iteration steps.



(a) 6 steps (b) 9 steps (c) 22 steps

Figure 5.3: Approximation of the stability region of (5.4.8).

5.4.2 Example B

Next, we consider the following DDE which is taken from [113]:

$$\dddot{x}\,(t) + A\dot{x}(t) + Bx(t-1) = 0 \tag{5.4.9}$$

Here, condition (5.2.5) has to be checked using the characteristic function

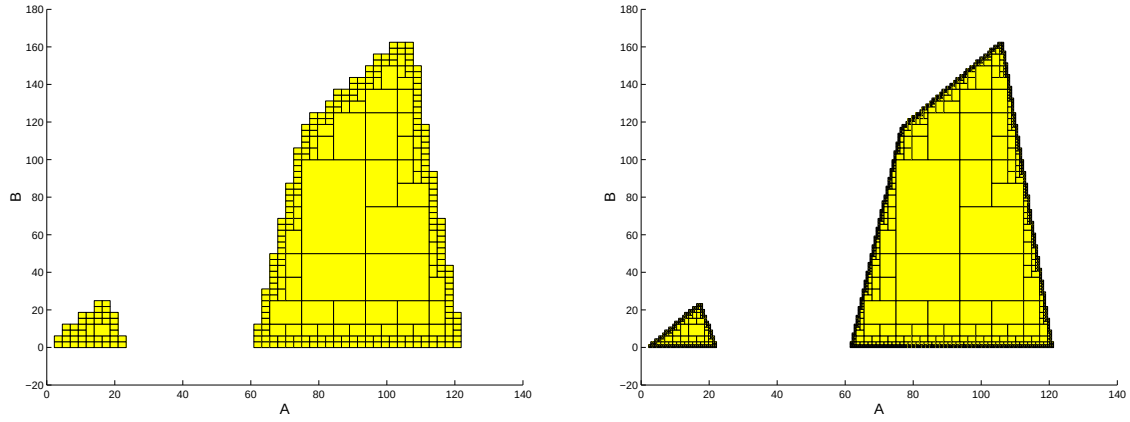
$$\Delta_2(z) = z^3 + Az + Be^{-z} = 0.$$

Figure 5.4 and Figure 5.5 show the approximated stability region for the delays $\tau = 1$ and $\tau = 2$. In a next step one may ask how these two approximations are connected under variation of τ. For this, the delay τ can be viewed as another free parameter of (5.4.9). The result can be seen in Figure 5.6.

5.4.3 Example C

Finally we consider the motivating stick balancing problem of the beginning of this chapter. We refer to [113] and [55] for more information about this example, e.g. the derivation of the underlying model:

$$\ddot{x}(t) - \frac{6g}{l}x(t) + \frac{6}{ml}A\dot{x}(t-\tau) + \frac{6}{ml}Bx(t-\tau) = 0 \tag{5.4.10}$$



(a) 10 steps (b) 15 steps

Figure 5.4: Two coverings of the stability region of Example B for the delay $\tau = 1$.

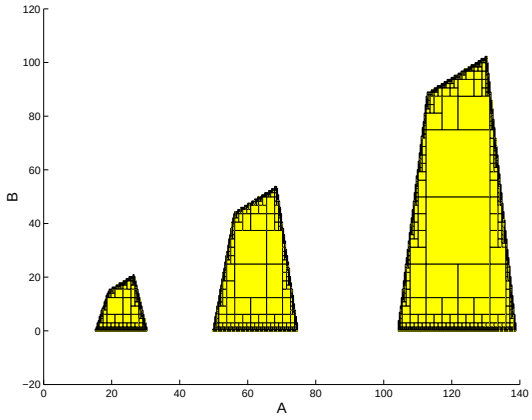


Figure 5.5: Approximated stability region for Example B and for the delay $\tau = 2$.

Figure 5.7 shows the stability regions within the domain

$$(\tau, A, B) \in [0, 0.3] \times [0, 10] \times [0, 30],$$

where the constants were chosen as follows:

$$m = 1, \quad l = 1, \quad g = 9.81.$$

It can be seen that there is no chance to balance the stick for someone whose reaction is slower than $\tau = 0.14$ – assuming this model to be realistic enough.

5.5 Conclusion

In this chapter we have presented a set oriented method for the approximation of arbitrary subsets $\mathcal{S} \subset Q$ of a given domain $Q \subset \mathbb{R}^n$. Furthermore, we have

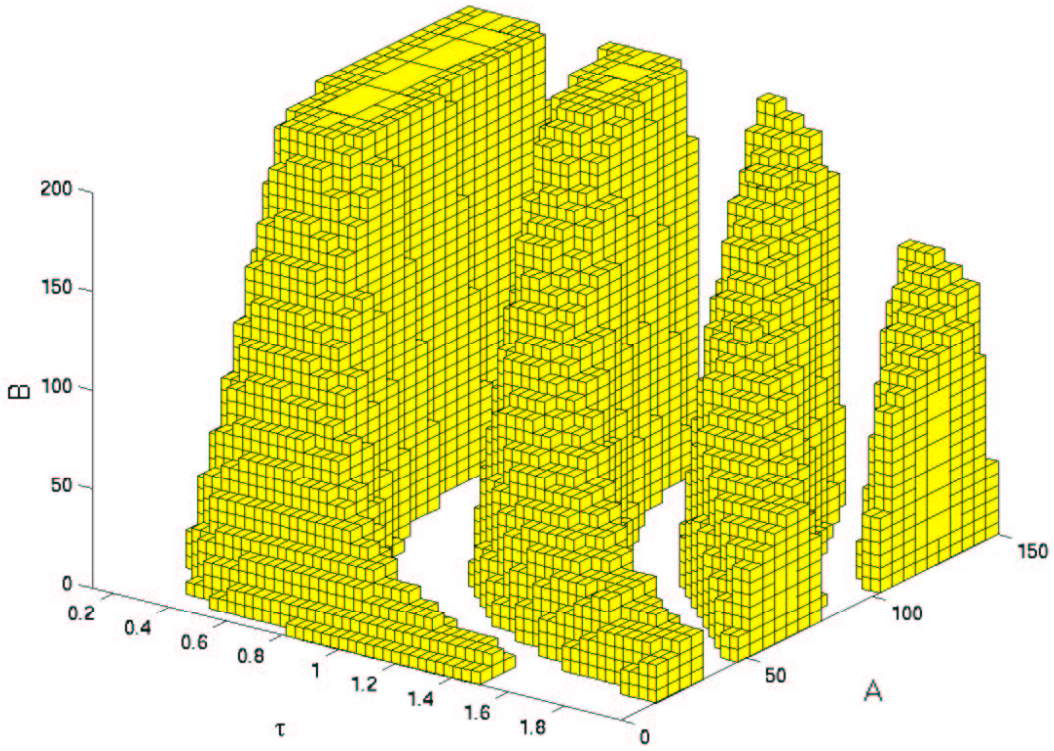


Figure 5.6: Covering of the stability region after 14 steps for Example B and $\tau \in [0, 2]$.

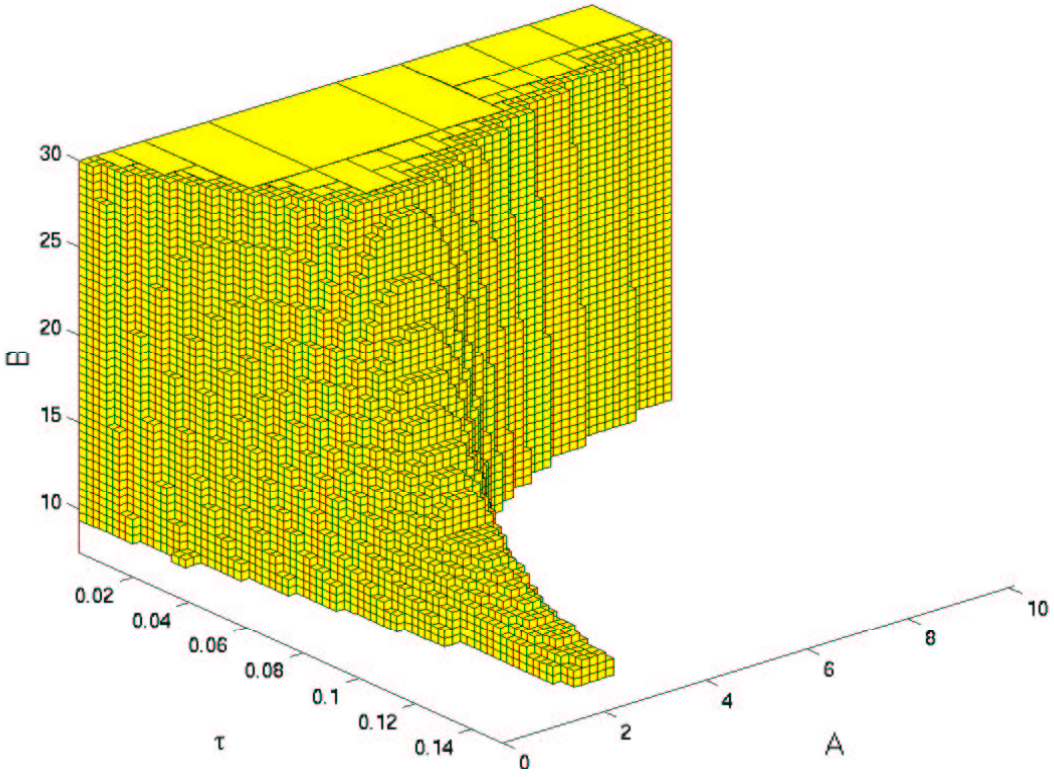


Figure 5.7: Stability region of the stick balancing problem (Example C).

demonstrated its applicability on the computation of stability regions of parameter dependent delay differential equations by some examples. Since no particular structure of the underlying characteristic function has to be taken into account, the method can be applied on a broad class of models.

Acknowledgement The author wants to thank Dr. Qinghua Zheng for calling his attention to this interesting field.

Chapter 6

Multi–Objective Optimization

6.1 Introduction

In a variety of applications in industry and finance a problem arises that several objective functions have to be optimized concurrently. For instance, for a perfect economical production plan, the ultimate desire would be to simultaneously *minimize cost* and *maximize quality*. This example already illustrates a natural feature of these problems namely that the different objectives typically contradict each other and therefore certainly do not have identical optima. Thus, the question arises how to approximate a particular "optimal compromise" (see e.g. [78] for an overview of widely used *interactive methods*) or how to compute *all* optimal compromises of this *multi–objective optimization problem* (MOP). The latter will be the topic of this chapter.

To get a first impression about multi–objective optimization and the corresponding decision–making problem we start with an example:
Let us consider that a customer is interested in buying a new motorcycle. More concretely, the customer is interested in a bike which should be fast and cheap. This leads directly to the optimization of *both* the cost and the maximum speed of a motorcycle in the (finite) set of available motorcycles. Let us assume that a shop offers (amongst each others) the motorcycles A, B and C as shown in Figure 6.1. The motorcycles A and B are not comparable according to the underlying optimization problem because bike A is (much) slower than bike B but on the other hand it is less expensive. Bike C is immediately discarded, because, according to the optimization model, it is slower and more expensive than bike B (C is said to be *dominated* by B). Hence the optimal motorcycles are bikes A and B. We will see later that this situation is typical for multi–objective optimization because the solution is not given by just one element but by a whole set of *optimal compromises*. When this set is detected, the problem arises to pick one element of this set – in the present example the customer is interested in buying *one* motorcycle. This *decision making problem* can not be solved in general but has to be done case by case. In the current motorcycle buying problem a decision could be made with the *additional* information that the customer is willing to spend a given budget for the motorcycle.

But in this case the problem can also be attacked by solving a scalar optimization problem (with proper constraints).

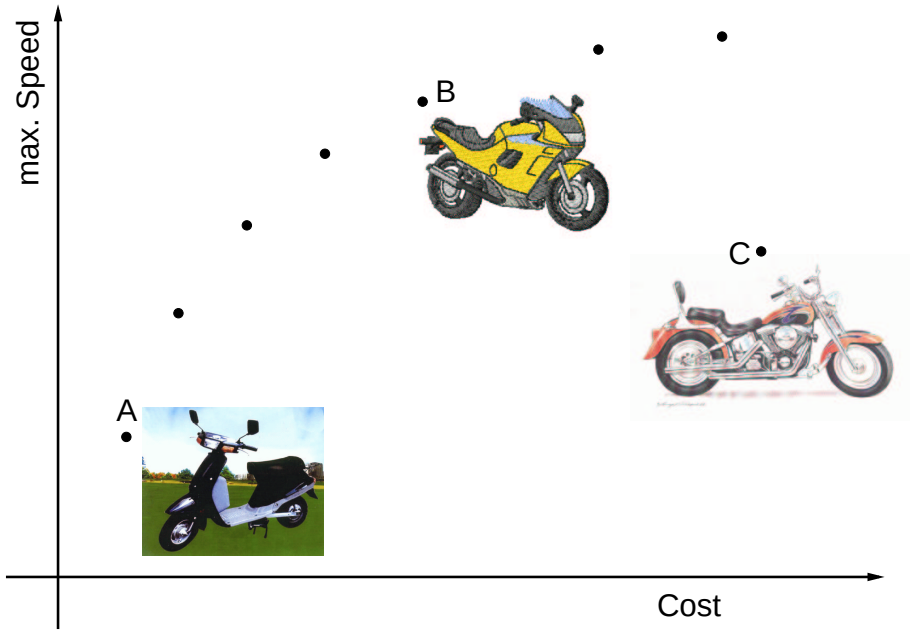


Figure 6.1: Hypothetical candidates for a possible motorcycle–buying decision–making problem.

Mathematically speaking in an MOP there are given k objective functions $f_1, \ldots, f_k : \mathbb{R}^n \to \mathbb{R}$ which have to be minimized. The set of optimal compromises with respect to the objective functions is called the *Pareto set*[1]. A point $x \in \mathbb{R}^n$ in parameter space is said to be a *Pareto point* if there is no other point which is at least as good as x in all the objectives and strictly better in at least one objective. Thus, the numerical solution of an MOP consists of an approximation of the Pareto set.

Multi–objective optimization is currently a very active area of research. By far most of the methods for the computation of single Pareto points or the entire Pareto set are based on a "scalarization" of the MOP (see e.g. [104] [105] [49] [85] [115] [18] and [43]). For a survey of these and further methods we refer to [78] and [58] for nonlinear MOPs and to [65] and [114] in the linear case.
Another way to attack the problem is by using *bio-inspired heuristics* like *Evolutionary Algorithms* (see [125] [22] [23] [21] [124] [45] [15]) or *Particle Swarm Optimization* (see [14] [40] [83] [82]). These methods are particularly advantageous in the situation where the MOP is discrete.
A method which is based on a stochastic approach is presented in [103]. In this work the authors derive a stochastic differential equation (SDE) which has the property that it is very likely to observe corresponding solutions in a neighborhood of the set of (local) Pareto points. Similar to the bio-inspired strategies here the idea is to

[1]Named after the economist Vilfredo Pareto, 1848-1923.

directly approximate the entire solution set and not just single points on the solution set.
Typically – that is under mild regularity conditions – the set of Pareto points is locally a $(k-1)$-dimensional manifold if there are k smooth objective functions. *Continuation methods* can be used to compute these solution sets efficiently. However, it has to be mentioned that these techniques are of local nature: given an initial set $\mathcal{S}_0$ of (local) Pareto points, all further solutions computed by these methods are restricted to the connected components of the set of (local) Pareto points which contain a point $s \in \mathcal{S}_0$. Continuation methods have been thoroughly analyzed over the last three decades, see e.g. [98], [2] and [56] for the computation of general implicitly defined manifolds. In the context of multi-objective optimization these techniques have for instance been used in [50], [96] and [58].

In this chapter we propose two numerical methods (i.e. subdivision techniques and continuation methods) for the computation of Pareto sets of given MOPs. Further, we will consider different smoothness assumptions on the underlying objective functions. We will start considering unconstrained MOPs but will further on also take (equality) constraints into account.
The outline of this chapter is as follows: first we will give the required theoretical background. Then we will propose three basic algorithms which can be used on general MOPs followed by extensions both for non–smooth and for smooth objectives leading to particular continuation methods. Numerical results will be presented after each of the three parts.

6.2 Theoretical Background

To succinctly summarize the theoretical background which is necessary for understanding our considerations in the subsequent sections, there are two different mathematical topics we have to address: the concept of Pareto optimality and the convergence toward Pareto sets by the subdivision algorithm.

6.2.1 Pareto Optimality

Definition of Pareto Optimality

In classical scalar optimization problems one has to find the (global) minimizer of a single real valued function $f : \mathbb{R}^n \to \mathbb{R}$. In a *multi–objective optimization problem* (MOP) the task is to simultaneously optimize k *objective functions* $f_1, \ldots, f_k : \mathbb{R}^n \to \mathbb{R}$. More precisely a general MOP can be stated as follows:

$$\min_{x \in R} \{F(x)\}, \quad R := \{x \in \mathbb{R}^n \,|\, h(x) = 0,\, g(x) \leq 0\}, \tag{MOP}$$

where the function F is defined as the vector of the objective functions

$$F : \mathbb{R}^n \to \mathbb{R}^k, \qquad F(x) = (f_1(x), \ldots, f_k(x)),$$

and $h : \mathbb{R}^n \to \mathbb{R}^m$, $m \leq n$, and $g : \mathbb{R}^n \to \mathbb{R}^q$. Obviously we have to define what is meant by finding the minimum of a vector valued function in (MOP). For this we state the following definition.

DEFINITION 6.2.1 (a) Let $v, w \in \mathbb{R}^k$. Then the vector v is *less than* w ($v <_p w$), if $v_i < w_i$ for all $i \in \{1, \ldots, k\}$. The relation $\leq_p$ is defined in an analogous way.

(b) A vector $v \in \mathbb{R}^k$ is *dominated* by a vector $w \in \mathbb{R}^k$ if $w \leq_p v$ and $v \neq w$ (i.e. there exists a $j \in \{1, \ldots, k\}$ such that $w_j < v_j$).

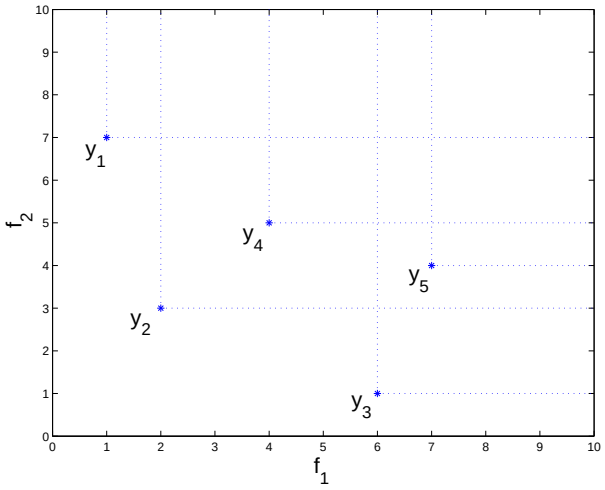


Figure 6.2: Dominated and non–dominated solutions: solution y_2 dominates y_4 and y_5. The solutions y_1 and y_2 (as well as e.g. y_2 and y_3) cannot be compared, i.e. y_1 and y_2 (as well as y_2 and y_3) are mutually non–dominating.

Since $\leq_p$ just defines a partial order on $\mathbb{R}^n$, we cannot proceed as in the classical scalar case. In fact, one cannot expect to find isolated stationary points. Rather one has to find the set of "optimal compromises" and – following Pareto ([90]) – these are defined in the following way.

DEFINITION 6.2.2 (a) Consider the multi–objective optimization problem (MOP). Then a point $\bar{x} \in R$ is called *(globally) Pareto optimal* or a *(global) Pareto point* if there is no $y \in R$ such that

$$F(y) \neq F(\bar{x}) \quad \text{and} \quad F(y) \leq_p F(\bar{x}). \tag{6.2.1}$$

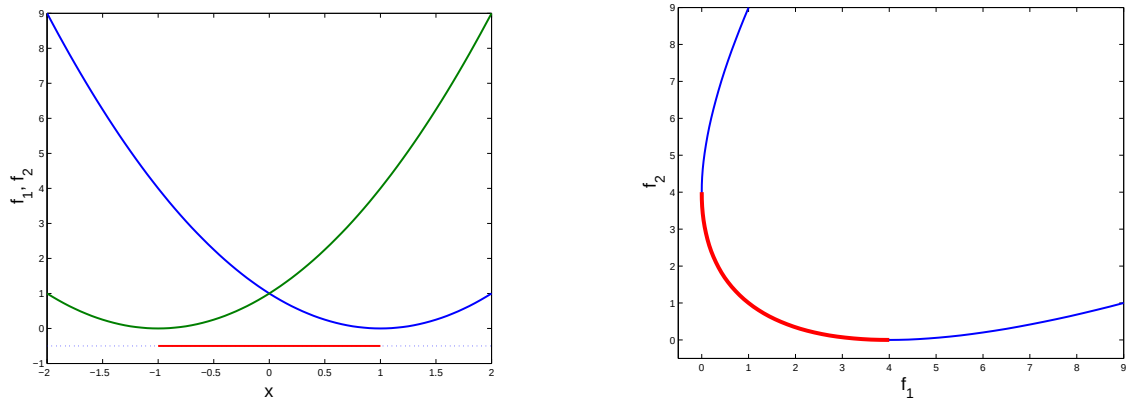
(b) A point $\bar{x} \in R$ is a *local Pareto point*, if there is a neighborhood $U(\bar{x}) \subset R$ of $\bar{x}$ such that there is no $y \in U(\bar{x})$ satisfying (6.2.1).

As a first example let us consider the following trivial MOP. Let F be given by $F(x) = (f_1(x), f_2(x))$, where

$$\begin{aligned} f_1, f_2 &: \mathbb{R} \to \mathbb{R} \\ f_1(x) &= (x-1)^2 \\ f_2(x) &= (x+1)^2 \end{aligned} \tag{6.2.2}$$

.

The Pareto set of MOP (6.2.2) is given by $\mathcal{P} = [-1, 1]$ which is the (one–dimensional) connection between the minimizers of the objectives f_1 and f_2. In Figure 6.3 the optimization problem is visualized in parameter space as well as in image space. Figure 6.3 (b) shows the classical shape of a Pareto set of a bicriteria problem (i.e. $k = 2$).



(a) Parameter Space

(b) Image Space

Figure 6.3: Multi–objective problem (6.2.2) and Pareto set in parameter space and in image space, indicated by the red line.

In this chapter we develop set oriented numerical methods for the approximation of the set of global Pareto points. Let us illustrate the notion of (global and local) Pareto points by the following example.

EXAMPLE 6.2.3 In Figure 6.4 we present an example of two objective functions $f_j : \mathbb{R} \to \mathbb{R}$, $j = 1, 2$. In this case the set of local Pareto points consists of the union of the intervals $[0, 1]$ and $[1.5, 2]$. However, only the points in the interval $[1.5, 2]$ are also globally Pareto optimal.

A Necessary Condition for Optimality

In our numerical methods we are going to make use of the following theorem of Kuhn and Tucker ([74]) which states a necessary condition for Pareto optimality for

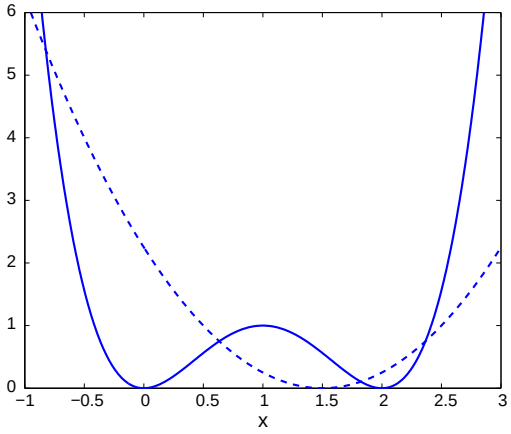


Figure 6.4: Two objective functions $f_j : \mathbb{R} \to \mathbb{R}$ $(j = 1, 2)$ on the interval $[-1, 3]$.

MOPs with equality constraints. For a more general formulation of the theorem we refer to [78] or [48].

THEOREM 6.2.4 *Let x^* be a Pareto point of (MOP) with $q = 0$. Let furthermore be the set of vectors $\{\nabla h_i(x) \,|\, i = 1, \ldots, m\}$ linearly independent. Then there exist vectors $\lambda \in \mathbb{R}^m$ and $\alpha \in \mathbb{R}^k$ with $\alpha_i \geq 0, i = 1, \ldots, k$ and $\sum_{i=1}^k \alpha_i = 1$ such that*

$$\begin{aligned} \sum_{i=1}^{k} \alpha_i \nabla f_i(x^*) + \sum_{j=1}^{m} \lambda_j \nabla h_j(x^*) = 0 \\ h_i(x^*) = 0, \; i = 1, \ldots, m. \end{aligned} \tag{6.2.3}$$

In the unconstrained case – i.e. for $m = 0$ – the theorem claims that the vector of zeros can be posed as a convex combination of the gradients of the objectives at every Pareto point. Obviously (6.2.3) is not a sufficient condition for (local) Pareto optimality. On the other hand points satisfying (6.2.3) are certainly "Pareto candidates" and thus, following [78], we now emphasize their relevance by the following

DEFINITION 6.2.5 A point $x \in \mathbb{R}^n$ is called a *substationary point* or *Karush–Kuhn–Tucker point*[2] (KKT–point) if there exist scalars $\alpha_1, \ldots, \alpha_k \geq 0$ and $\lambda \in \mathbb{R}^m$ such that (6.2.3) is satisfied.

Qualitative Description of the Pareto Set

Having stated the Theorem 6.2.4, one is in the position to give a qualitative description of the set of Pareto optimal solutions.

[2]Named after the works of Karush [68] and Kuhn & Tucker [74] for scalar–valued optimization problems.

Denote by $\tilde{F} : \mathbb{R}^{n+m+k} \to \mathbb{R}^{n+m+1}$ the following auxiliary function:

$$\tilde{F}(x, \lambda, \alpha) = \begin{pmatrix} \sum_{i=1}^{k} \alpha_i \nabla f_i(x) + \sum_{j=1}^{m} \lambda_j \nabla h_j(x^*) \\ h(x) \\ \sum_{i=1}^{k} \alpha_i - 1 \end{pmatrix} \tag{6.2.4}$$

By Theorem 6.2.4 it follows that for every KKT–point $x^* \in \mathbb{R}^n$ there exist vectors $\lambda^* \in \mathbb{R}^m$ and $\alpha^* \in \mathbb{R}^k$ such that

$$\tilde{F}(x^*, \lambda^*, \alpha^*) = 0. \tag{6.2.5}$$

Hence one expects that the set of KKT–points defines a $(k-1)$-dimensional manifold due to the Implicit Function Theorem. This is indeed the case under certain smoothness assumptions, see [58] for a thorough discussion of this topic.

A Descent Direction

Similar to classical iteration schemes for the numerical solution of scalar optimization problems we would like to identify a descent direction for our numerical methods. More precisely, for a point which is not substationary we need to know a direction in $\mathbb{R}^n$ in which all the k objectives are simultaneously decreasing. For this purpose we briefly summarize in the following the main results of [103], which we will use for our computations[3]. These results have to be viewed as the natural extension of the classical underlying theory for the method of steepest descent.

First we associate with $F : \mathbb{R}^n \to \mathbb{R}^k$, $F(x) = (f_1(x), \ldots, f_k(x))$, the following quadratic optimization problem:

$$\min_{\alpha \in \mathbb{R}^k} \left\{ \left\| \sum_{i=1}^{k} \alpha_i \nabla f_i(x) \right\|_2^2 ; \alpha_i \geq 0, i = 1, \ldots, k, \sum_{i=1}^{k} \alpha_i = 1 \right\} \tag{QOP}$$

Then one can show the following result.

THEOREM 6.2.6 ([103]) *Let $q : \mathbb{R}^n \to \mathbb{R}^n$ be defined by*

$$q(x) = \sum_{i=1}^{k} \hat{\alpha}_i \nabla f_i(x),$$

where $\hat{\alpha}$ is a solution of (QOP). Then

(i) either $q(x) = 0$ or $-q(x)$ is a descent direction for all objective functions $f_1, \ldots, f_k$ in x;

[3] Alternatively, other descent directions can be used, for example the ones proposed in [44].

(ii) for each $\hat{x} \in \mathbb{R}^n$ there is a neighborhood $U(\hat{x})$ and a constant $L_{\hat{x}} \geq 0$, such that

$$\|q(x) - q(y)\|_2 \leq L_{\hat{x}} \|x - y\|_2 \quad \textit{for all} \quad x, y \in U(\hat{x}).$$

By this result the initial value problem

$$\dot{x}(t) = -q(x(t)), \quad x(0) = x_0, \tag{6.2.6}$$

is well posed. Let $x : [0, \infty[\to \mathbb{R}^n$ be the unique solution of (6.2.6). Then one can show that this solution satisfies

$$F(x(s)) \geq_p F(x(t)) \quad \text{and} \quad F(x(s)) \neq F(x(t)) \qquad \text{for all} \quad 0 \leq s < t < \infty.$$

In fact, suppose additionally that the set

$$R_{\leq_p} = \{x \in \mathbb{R}^n : F(x) \leq_p F(x_0)\}$$

is bounded. Then the solution $x(t)$ has to converge to a substationary point for $t \to \infty$. Thus, a suitable discretization of the "generalized gradient system" (6.2.6) yields numerical iteration schemes converging towards substationary points.

Observe that by Theorem 6.2.6 each Pareto point is a zero of the function q. Therefore the aim is to find the set of zeros of q. Since these zeros are not isolated a set oriented approach turns out to be most adequate for their approximation.

6.2.2 Convergence toward Pareto Sets

By Theorem 6.2.6 each Pareto point is a zero of the function q as defined in the same theorem. In analogy to classical zero finding procedures with different step size strategies we now prove that an appropriate discretization of the ordinary differential equation (6.2.6) will lead to iteration schemes which generate sequences converging towards substationary points. Here we essentially proceed along the lines of [32]. Then we can conclude – using the convergence result in Proposition 2.2.7 – that an application of the subdivision algorithm yields a close covering of the set of substationary points.

In order to simplify the notation we begin by defining for a nonnegative vector $b \in [0, 1]^k$ the corresponding objective function (cf. (MOP))

$$F_b(x) = b^T F(x).$$

We assume that the derivative of each F_b is Lipschitz continuous with (uniform) Lipschitz constant L.

We now discretize (6.2.6) and consider the following iteration scheme

$$x_{j+1} = x_j + h_j p_j, \quad j = 0, 1, \ldots. \tag{6.2.7}$$

In the following we will sometimes denote the right hand side by P, i.e.

$$P(x_j) = x_j + h_j p_j, \quad j = 0, 1, \ldots. \tag{6.2.8}$$

The descent direction p_j is chosen such that for positive constants $\sigma, \tau > 0$

$$-\frac{q(x_j)^T p_j}{\|q(x_j)\|\|p_j\|} \geq \sigma \quad \text{and} \quad \|p_j\| \geq \tau \|q(x_j)\|,$$

where $q(x_j)$ is defined in Theorem 6.2.6. The step size h_j is an Armijo or a Powell step size. Let x_0 be the initial point and suppose that the iteration can be perfomed for $j \to \infty$ such that the sequence $(x_j)_{j=0,1,\ldots}$ lies within a compact set $D \subset \mathbb{R}^n$.

PROPOSITION 6.2.7 *Suppose that x^* is an accumulation point of the sequence $(x_j)_{j=0,1,\ldots}$. Then x^* is a substationary point for the multi–objective optimization problem (MOP).*

Proof: First observe that if one of the x_j's is a substationary point then we are done (cf. Theorem 6.2.6).

Thus, without loss of generality we may assume that $x_j \to x^*$ for $j \to \infty$ and that none of the x_j's is substationary. Consider the corresponding sequence $a_j = (\alpha_1^j, \ldots, \alpha_k^j)$ of solution vectors of the optimization problem (QOP) in step j of the iteration procedure. Since $\alpha_1^j, \ldots, \alpha_k^j \in [0, 1]$ we may assume that $a_j \to a$ for $j \to \infty$. (Otherwise restrict the following considerations to a subsequence.)

We now show that the sequence (x_j) converges to a stationary point for $F_a(x)$ and, thus, proving the desired result.

Using the fact that h_j is an Armijo or a Powell step size in x_j we have by classical results on iteration schemes for optimization problems (see e.g. [32]) that there exists a constant $\theta > 0$ such that

$$F_{a_j}(x_j) - F_{a_j}(x_{j+1}) \geq \theta \min \left[-\nabla F_{a_j}(x_j)^T p_j, \left(\frac{\nabla F_{a_j}(x_j)^T p_j}{\|p_j\|} \right)^2 \right] \tag{6.2.9}$$

in each step of the iteration process. Observe that θ does not depend on j by the assumption on the uniform Lipschitz continuity of ∇F_b.

Now suppose that

$$F_a(x_j) - F_a(x_{j+1}) < \theta \min \left[-\nabla F_a(x_j)^T p_j, \left(\frac{\nabla F_a(x_j)^T p_j}{\|p_j\|} \right)^2 \right] \tag{6.2.10}$$

for infinitely many j. By our assumption on the descent direction we have

$$-\frac{\nabla F_a(x_j)^T p_j}{\|\nabla F_a(x_j)\|\|p_j\|} \geq \frac{\sigma}{2} \quad \text{and} \quad \|p_j\| \geq \frac{\tau}{2} \|\nabla F_a(x_j)\|$$

for all $j \geq j_0$. Combining these estimates with (6.2.9) and (6.2.10) we obtain

$$0 = \lim_{j \to \infty} \min \left[-\nabla F_a(x_j)^T p_j, \left(\frac{\nabla F_a(x_j)^T p_j}{\|p_j\|} \right)^2 \right] \geq \frac{\theta \sigma}{4} \min(\tau, \sigma) \|\nabla F_a(x^*)\|^2$$

and x^* is substationary as desired.

It remains to consider the case where

$$F_a(x_j) - F_a(x_{j+1}) \geq \theta \min \left[-\nabla F_a(x_j)^T p_j, \left(\frac{\nabla F_a(x_j)^T p_j}{\|p_j\|} \right)^2 \right]$$

for all $j \geq j_1$. Here we obtain in an analogous way

$$F_a(x_j) - F_a(x_{j+1}) \geq \frac{\theta\sigma}{4} \min(\tau, \sigma) \|\nabla F_a(x_j)\|^2 \geq 0$$

for all $j \geq \max(j_0, j_1)$. Letting $j \to \infty$ it follows that $\|\nabla F_a(x^*)\| = 0$. □

For the remainder of this section we now assume that $p_j = q(x_j)$ and that we are working with s different step sizes h_ℓ. Then, combining Propositions 6.2.7 and 2.2.7 we immediately obtain the following result:

COROLLARY 6.2.8 *Suppose that the set $\mathcal{S}$ of substationary points is bounded and let D be a compact neighborhood of $\mathcal{S}$. Then an application of the subdivision algorithm to D with respect to the iteration scheme (6.2.7) creates a covering of the entire set $\mathcal{S}$, that is,*

$$\mathcal{S} \subset Q_k \quad \text{for } k = 0, 1, 2, \ldots$$

in the course of the subdivision process.

Observe that we have shown that the covering obtained by the subdivision process becomes "tight". However we cannot prove convergence towards $\mathcal{S}$ without an additional assumption on its structure. We illustrate this fact by the following example.

EXAMPLE 6.2.9 Let us reconsider Example 6.2.3 (cf. Figure 6.4). In that case an application of the subdivision algorithm to the interval $D = [-1, 3]$ will converge to the interval $[0, 2]$. Thus, we obtain a covering of the set $\mathcal{S} = [0, 1] \cup [1.5, 2]$ but we also approximate the additional part $(1, 1.5)$.

For a proof of this fact one has to observe that a box which contains the number 1 as well as points which are bigger than 1 always has a nonzero intersection with its image under the iteration scheme (6.2.7). Moreover the image of this box also has a nonzero intersection with its right neighbor. Proceeding with this neighboring box we see that all the boxes between 1 and 1.5 have preimages in other boxes in each step of the subdivision process. Therefore the interval $(1, 1.5)$ is never removed in the selection step.

We will see in Section 6.3 how to overcome the problem described in the previous example in actual realizations of the algorithm. However, these considerations in combination with standard compactness arguments immediately lead to the following convergence result:

COROLLARY 6.2.10 *Suppose that the set $\mathcal{S}$ of substationary points is bounded and connected. Let D be a compact neighborhood of $\mathcal{S}$. Then an application of the*

subdivision algorithm to D with respect to the iteration scheme (6.2.7) leads to a sequence of coverings which converges to the entire set $\mathcal{S}$, that is,

$$h(\mathcal{S}, Q_k) \to 0 \quad \textit{for } k = 0, 1, 2, \ldots,$$

where h denotes the usual Hausdorff distance.

The following remark addresses a straightforward but interesting consequence of this result.

REMARK 6.2.11 Under the assumptions of Corollary 6.2.10 we can conclude – in the unconstrained case – that the set $\mathcal{S}$ has to have trivial homotopy. For instance, in case of two objective functions on a (at least) two–dimensional parameter space the set $\mathcal{S}$ cannot be topologically equivalent to a circle.

6.3 Basic Algorithms

In this section we propose three different algorithms for the computation of the Pareto set of a given MOP, or, to be more precise, we present algorithms for the computation of tight coverings of such sets. Moreover we propose some guidelines on how to combine these algorithms in order to increase the performance of the respective numerical schemes.

6.3.1 Subdivision Algorithm

The first algorithm is directly based on the theoretical considerations of the previous section, in particular on Corollaries 6.2.8 and 6.2.10. In fact, we now discuss a concrete realization of the subdivision procedure for the computation of tight coverings of the set of substationary points using the dynamical system

$$x_{j+1} = P(x_j) = x_j + h_j p_j, \quad j = 0, 1, \ldots \ .$$

Descent Direction In all the computations of unconstrained MOPs presented in this chapter we have used the descent direction

$$p_j = q(x_j),$$

cf. (6.2.7) and Theorem 6.2.6.

For the specific case of (unconstrained) bicriteria optimization problems (i.e. $k = 2$) one could alternatively use the following descent direction:

$$p_j = -\left(\frac{\nabla f_1(x_j)}{\|\nabla f_1(x_j)\|_2} + \frac{\nabla f_2(x_j)}{\|\nabla f_2(x_j)\|_2} \right).$$

This choice is particularly useful in the case where the cost for the evaluation of $\nabla f_i, \ i = 1, 2$, is high. We have tested this descent direction with all the differentiable bicriteria optimization problems presented in this chapter yielding satisfying results.

Step Length Following standard techniques for step length control – see e.g. [32] – we have chosen a particular Armijo step size strategy in the following way: starting with the given point x_j we evaluate F along the descent direction p_j in uniform step lengths h_0 as long as the values of all objectives decrease. Once one objective function starts to increase, a "better" iterate x_{j+1} with intermediate step length is calculated via backtracking:

(i) $n := 1$

(ii) **while** $F(x_j + nh_0p_j) <_p F(x_j + (n-1)h_0p_j)$ **and**
$\langle \nabla f_i(x_j + (n-1)h_0p_j), p_j \rangle < 0 \quad \forall i = 1, \ldots, k$
set $n := n + 1$

(iii) **choose** $x_{j+1} \in [x_j + (n-1)h_0p_j,\ x_j + nh_0p_j]$
such that $F(x_{j+1}) <_p F(x_j + (n-1)h_0p_j)$

REMARKS 6.3.1 (a) To find an appropriate guess for the "scanlength" h_0 it is possible to take advantage of the subdivision scheme. If a step length $\tilde{h}$ has been computed for a point $\tilde{x}$ inside a certain box then this distance can be chosen as the scanlength for the following points inside the same box. This strategy works particularly well when the subdivision scheme is at a level where all the boxes are already quite small.

(b) Finally we propose the following backtracking procedure in step (2) above. For every $i \in \{1, \ldots, k\}$ with $f_i(x_j + nh_0p_j) > f_i(x_j + (n-1)h_0p_j)$ we determine

$$x_j^i = x_j + ((n-1)h_0 + \Theta_i)p_j, \quad \Theta_i \in (0, 1),$$

via quadratic backtracking. Then set

$$\hat{x}_j = x_j^i \quad \text{where } i \text{ is chosen such that } \Theta_i \text{ is minimal.}$$

If $F(\hat{x}_j) <_p F(x_j + (n-1)h_0p_j)$ then the point $\hat{x}_j$ is accepted and we choose $x_{j+1} = \hat{x}_j$ in step (2). Otherwise proceed in the same way to find a new iterate between $x_j + (n-1)h_0p_j$ and $\hat{x}_j$.

It should be mentioned that the subdivision algorithm (in the following denoted by DS–Subdivision) could be made reliable if the Lipschitz estimates of the underlying dynamical system P would be taken into account, using e.g. the methods described in [66]. In particular in the present case where P is given by a discretization of an ODE of the following form:

$$x_{i+1} = P(x_i) = x_i + hG(x_i).$$

If G is Lipschitz continuous with Lipschitz constant $\mathcal{L}_G$, also P is Lipschitz continuous with constant $\mathcal{L}_P$, where

$$\mathcal{L}_P = 1 + h\mathcal{L}_G.$$

The rigorous computation of the attractor of P (i.e. the set $\mathcal{S}$ of substationary points) as described in [66] would lead to two possiblities in case $\mathcal{L}_G$ is large: (a) a lot of test points have to be evaluated (in particular in higher dimensions) or (b) the estimate $\mathcal{L}_P$ could be reduced by the choice of a small steplength h. But also this would increase the computational time significantly due to the sparse progress of the sequences $\{x_i\}$, and hence a growth of the number of boxes of the collections – that is anyway large (see Section 2.4) – is expected. Thus, we have to find other possible ways for the satisfying computation of the Pareto set. One possibility is proposed in the next section.

6.3.2 Recovering Algorithm

It may be the case that in the course of the subdivision procedure boxes get lost although they contain substationary points. This will for instance be the case when there are not enough test points taken into account for the evaluation of $F(B)$ for a box $B \in \mathcal{B}_k$ (see Section 2.3). We now describe an algorithm using a kind of "healing" process which allows to recover those substationary points which have previously been lost.

Before we can state the algorithm we have to present some more technical details about box collections. For theoretical purposes denote by $\mathcal{P}_k$ a *complete* partition of the set $Q = B_{\hat{c},\hat{r}}$ into boxes of subdivision size – or *depth*[4] – k, which are generated by successive bisection of Q. Then there exists for every point $y \in Q$ and every depth k exactly one box $B(y,k) \in \mathcal{P}_k$ with center c and radius r such that $c_i - r_i \leq y_i < c + r_i,\ \forall i = 1, \ldots, n$.

Figure 6.5: Recovering algorithm: uncomplete covering of the Pareto set (left) and possible choice of test points for a given box B (right).

The aim of the algorithm is to extend the given box collection step by step along the covered parts of the set $\mathcal{S}$ of substationary points until no more boxes are added.

[4] $\mathcal{P}_k$ and hence every box collection considered here can be identified with a set of leaves of a binary tree of depth k, see Section 2.3.

In order to find the corresponding neighboring boxes of a box B we take starting points $\{s_i\}_{i=1,\dots,l}$ near B and compute $\mathcal{X} = \{P^q(s_i)|i = 1,\dots,l\}$ with a suitable power $q \geq 1$ in order to obtain points both near B and $\mathcal{S}$. Afterwards the box collection is extended by the boxes $B \in \mathcal{P}_k$ which contain elements from $\mathcal{X}$. In the first step this is done for all boxes from the box collection, for the following steps this local search has to be performed only in the neighborhood of the boxes which were added in the preceding step.
For a given box collection $\mathcal{B}_k$ the algorithm reads as follows:

(i) **for all** $B \in \mathcal{B}_k$
  $B.active := TRUE$
(ii) **for** $i = 1,\dots,MaxStep$
  $\hat{\mathcal{B}}_k := \mathcal{B}_k$
  for all $\{B \in \mathcal{B}_k : B.active == TRUE\}$
    choose starting points $\{s_i\}_{i=1,\dots,l}$ near B
    $\mathcal{X} := \{P^q(s_i)|i = 1,\dots,l\}$
    $B.active := FALSE$
    for all $y \in \mathcal{X}$:
      if $B(y,k) \notin \mathcal{B}_k$
        $\mathcal{B}_k := \mathcal{B}_k \cup B(y,k)$
        $B(y,k).active := TRUE$
  if $\hat{\mathcal{B}}_k == \mathcal{B}_k$ **STOP**

Hence the recovering algorithm allows to add boxes to the given collection. The desired covering of the set $\mathcal{S}$ of substationary points cannot get worse but will improve if the parameters of the algorithm are adjusted properly. On the other hand, the recovering algorithm does not adequately perform in the case where a box does not contain part of $\mathcal{S}$ but is possibly far away. In this case the algorithm would extend the box covering by many undesired regions on the way towards $\mathcal{S}$ in the course of the iteration of test points. This observation is particularly valid for higher dimensional parameter spaces. A method which allows to overcome this problem can be found in Section 6.6.

Computational Effort Similar to the discussion made in Section 2.4 we can give (rough) estimates on the computational effort of the recover algorithm. Let us again assume we are given an m-dimensional[5] manifold $\mathcal{M}$ in n-dimensional space and furthermore we are given a box $B \in \mathcal{B}_k$ which contains some part of $\mathcal{M}$. Then we can assume that B has $2^m - 1$ neighbor boxes which also contain some part of $\mathcal{M}$, hence these boxes build up an m-dimensional cube if the boxes are small enough. By the same argument (and because $\mathcal{M}$ is locally diffeomorph to an m-dimensional cube) we have to expect to extend $\mathcal{B}_k$ by $5^3 - 1$ boxes around B after 2 recovering steps, whereby $5^3 - 3^3$ boxes were added in the last steps and will be "active" in step 3.
Starting with one single box which contains some part of $\mathcal{M}$, we can assume the

[5]In the context of multi–objective optimization it is $m = k - 1$.

following number of boxes in the first (few) iteration steps:

$$\begin{aligned} |\mathcal{B}_0| &= 1 \\ |\mathcal{B}_l| &= (2l-1)^m + \underbrace{(2l+1)^m - (2l-1)^m}_{\text{active boxes}} \end{aligned} \tag{6.3.11}$$

The expected number of boxes which are needed to cover $\mathcal{M}$ can be estimated by:

$$|\mathcal{B}_l| = (\sqrt[n]{2^m})^l,$$

where l is the insertion depth of the initial box B. For example, given an MOP $F : \mathbb{R}^3 \to \mathbb{R}^3$ and the insertion deph $l = 15$ there have to be taken approximately (at least) 1024 boxes in order to cover $\mathcal{M}$. For $l = 21$ about $1.6 \cdot 10^4$ and for $l = 30$ about $1.0 \cdot 10^7$ boxes seem to be required.

6.3.3 Sampling Algorithm

Observe that there are a couple of potential drawbacks which may occur when using the two algorithms described above:

(a) the gradients of the objectives are needed,

(b) the set $\mathcal{S}$ is generally a strict superset of the Pareto set, and

(c) the algorithms are capable of finding local Pareto points on the boundary of the domain Q – e.g. via penalization strategies. However, it turned out in practice that typically a large part of ∂Q is locally optimal (see e.g. Figure 6.6).

The following sampling algorithm avoids all these problems because it takes only the function values of the objective functions into account. On the other hand this algorithm is not as robust to errors as the first two ones because it is only global relative to the underlying box collection.

In the following we call a point $x \in \mathcal{X} \subset \mathbb{R}^n$ *nondominated* with respect to F and $\mathcal{X}$ if there does not exist a point $y \in \mathcal{X}$ with $F(y) \leq_p F(x)$ and $F(y) \neq F(x)$. Using this notion an outline of the algorithm is as follows. Given a box collection $\mathcal{B}_{k-1}$ the collection $\mathcal{B}_k$ is obtained by:

(i) **Subdivision**
Construct from $\mathcal{B}_{k-1}$ a new system $\hat{\mathcal{B}}_k$ of subsets such that

$$\bigcup_{B \in \hat{\mathcal{B}}_k} B = \bigcup_{B \in \mathcal{B}_{k-1}} B$$

and

$$\operatorname{diam}(\hat{\mathcal{B}}_k) = \theta_k \operatorname{diam}(\mathcal{B}_{k-1}),$$

where $0 < \theta_{min} \leq \theta_k \leq \theta_{max} < 1$.

(ii) **Selection**
for all $B \in \hat{\mathcal{B}}_k$

choose a set of **test points** $\mathcal{X}_B \subset B$

$N :=$ nondominated points of $\bigcup_{B \in \hat{\mathcal{B}}_k} \mathcal{X}_B$

$$\mathcal{B}_k := \left\{ B \in \hat{\mathcal{B}}_k : \exists y \in \mathcal{X}_B \cap N \right\}$$

The approach of this algorithm has similar characteristics to well known branch and bound strategies used for scalar optimization e.g. described in [62] or in [111] and [12]. However, in contrast to these algorithms we have omitted any bounding strategy. This can be done because of the special structure of the problem: the larger the number k of objectives is the more robust the selection step of the algorithm becomes. These observations coincide with our intuition concerning the nature of multi–objective optimization. For the efficient realization of the nondominance test in the selection step we basically use the data structure described in Section 6.5.

Figure 6.6: Examples of MOPs with optima relative to the boundary. Left: the point a is a Pareto point of the MOP given by $F(x) = (f_1(x), f_2(x))$ and $Q = [a, b]$. Right: a covering of the set of local Pareto points. For a detailed discussion of this particular MOP we refer to Section 6.4.4.

6.3.4 Usage and Combination of the Algorithms

In principle each of the algorithms proposed above is applicable to an MOP on its own. The *subdivision algorithm* has the advantage of being very robust with respect to errors by the use of the descent direction. On the other hand, all the gradients of the objectives have to be available and the algorithm is unable to distinguish between a local and a global Pareto point. Furthermore the efficiency of the algorithm will get worse when the MOP has optima relative to the boundary of the domain.

The *recovering algorithm* is able to extend the computed box covering of the set of substationary points but it is just local in nature.

The *sampling algorithm* is able to detect global Pareto points even on the boundary of the domain due to the fact that it works in the image space of the MOP. Naturally, there remains always uncertainty due to the sampling approach, in particular when the boxes are big and/or the dimension of the MOP is large. Nevertheless, results have shown that this algorithm works quite well, in particular when the gradients of the objectives are not available and the dimension of the MOP is moderate.

To obtain an even better performance – i.e. to compute a robust approximation of the Pareto set *and* to use a moderate amount of function calls – we propose the following combination of the three algorithms. Here we assume that the gradients of all objectives are available.

Step 1 Start with the subdivision algorithm because of its robustness. Take a few test points for the evaluation of the boxes via the dynamical system P.

Step 2 Apply the recovering algorithm to the box collection which has been computed in Step 1 in order to fill the gaps which have possibly been generated before.

Step 3 Use the sampling algorithm to tighten the extended box collection. By using this algorithm boxes which only contain local Pareto points can be removed from the covering. Furthermore boxes get removed which contain no substationary points but where kept in Step 1 because of the weak convergence of P in these regions.

There are of course other possible ways to combine the algorithms, e.g. it is possible to apply again the first algorithm on the box collection obtained by the procedure described above. In Step 2 the number of boxes which are added to the collection is a measure for the number of test points needed in Step 1.

6.4 Numerical Results for General Models

In this section we illustrate the efficiency of our basic set oriented algorithms for the computation of Pareto sets by several numerical examples.

6.4.1 Example G1

We begin by considering a simple example in order to illustrate the working principle of the subdivision procedure.

The MOP is given by two objective functions $f_1, f_2 : \mathbb{R}^2 \to \mathbb{R}$,

$$\begin{aligned} f_1(x_1, x_2) &= (x_1 - 1)^2 + (x_2 - 1)^4, \\ f_2(x_1, x_2) &= (x_1 + 1)^2 + (x_2 + 1)^2. \end{aligned} \tag{6.4.12}$$

The basic region in parameter space is chosen to be $Q = [-5, 5] \times [-5, 5]$. In Figure 6.7 we show box coverings generated by the subdivision algorithm after several steps indicating that these sets indeed converge to the set of Pareto points.

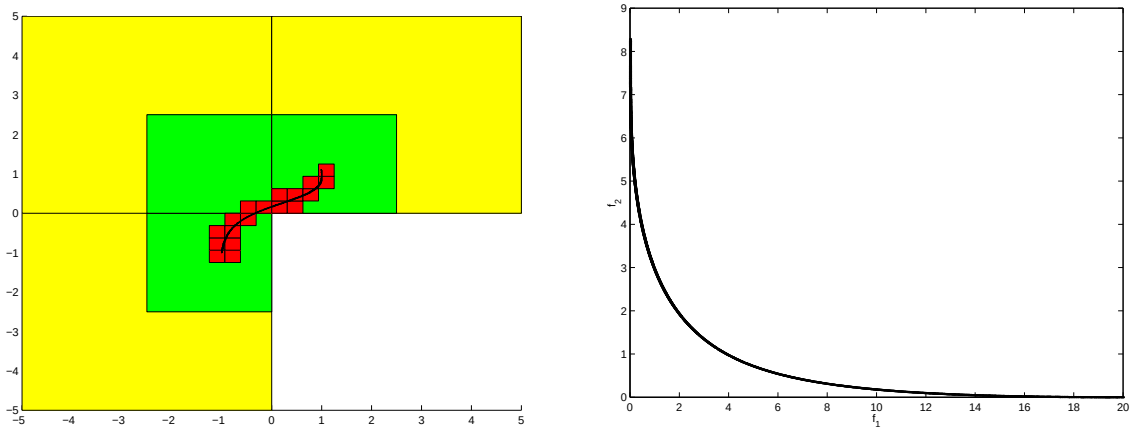


Figure 6.7: Successive approximations of the Pareto set of MOP (6.4.12). Left: $\mathcal{B}_2$ (yellow), $\mathcal{B}_4$ (green), $\mathcal{B}_{10}$ (red) and $\mathcal{B}_{20}$ (black); right: the image $F(\mathcal{B}_{20})$, i.e. an approximation of the Pareto optimal solutions in image space.

6.4.2 Example G2

The following example is taken from [58]:

$$\begin{aligned} f_1, f_2 : \mathbb{R}^2 &\to \mathbb{R} \\ f_1(x_1, x_2) &= \cos(a(x)) \cdot b(x) \\ f_2(x_1, x_2) &= \sin(a(x)) \cdot b(x) \\ \text{where} \quad a(x) := \frac{2\pi}{360}&(a_c + a_1 \cdot \sin(2\pi x_1) + a_2 \cdot \sin(2\pi x_2)) \\ b(x) &:= 1 + d \cdot \cos(2\pi x_1) \\ \text{with} \quad a_c = 45, a_1 = 40, a_2 = 25 \quad &\text{und} \quad d = 0.5. \end{aligned} \tag{6.4.13}$$

Since $F = (f_1, f_2)$ is periodic, it is sufficient to consider the domain $Q = [0, 1] \times [0, 1]$. The computed solution is shown in Figure 6.8. The covering of the set of Pareto points (left side) reveal two regions where the convergence is slow. But in fact

$$\text{Rank}(D_F((0.5, 0.25)^T) = \text{Rank}(D_F((0.5, 0.75)^T) = 1,$$

i.e. the set of Pareto points is not a smooth manifold. By the global approach of the subdivision algorithm it is practicable to calculate the whole set in one computation while for example at least three initial Pareto points have to be known to calculate the set using continuation methods, see [58].

6.4.3 Example G3

In this example we consider three objective functions $f_1, f_2, f_3 : \mathbb{R}^3 \to \mathbb{R}$, where

$$\begin{aligned} f_1(x_1, x_2, x_3) &= (x_1 - 1)^4 + (x_2 - 1)^2 + (x_3 - 1)^2, \\ f_2(x_1, x_2, x_3) &= (x_1 + 1)^2 + (x_2 + 1)^4 + (x_3 + 1)^2, \\ f_3(x_1, x_2, x_3) &= (x_1 - 1)^2 + (x_2 + 1)^2 + (x_3 - 1)^4. \end{aligned} \tag{6.4.14}$$

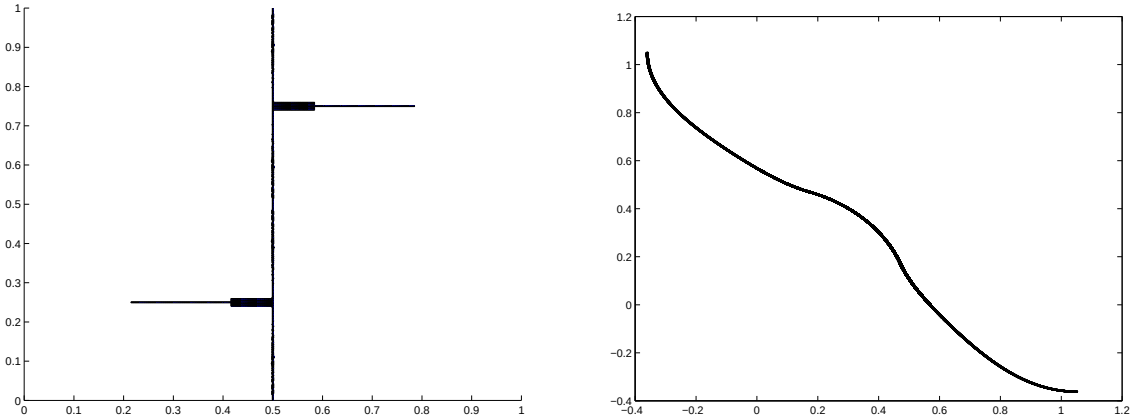


Figure 6.8: The box–collection $\mathcal{B}_{20}$ of MOP (6.4.13) (left) and its image (right).

The basic domain is chosen as $Q = [-5, 5]^3$. The resulting box collections are shown in Figures s 6.9 and 6.10.

Furthermore, this example can be taken to illustrate how the different algorithms can be combined in order to achieve a better performance. The result shown in Figure 6.11 was obtained by the following steps:
first the subdivision algorithm was applied for 21 steps using only the center point of every box as the test point for the dynamical system P (Figure 6.11(a)). Using only these few test points the computed box collection $\mathcal{B}_{21}$ already reveals the shape of the set of Pareto points but it also contains many holes. These holes could be filled by an application of the recovering algorithm on $\mathcal{B}_{21}$ (Figure 6.11(b)). Finally, the covering was tightened using the sampling algorithm.

6.4.4 Example G4

We now solve an MOP which serves as a model for a problem occuring in production planning (cf. [103]). Here we have two objective functions $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$,

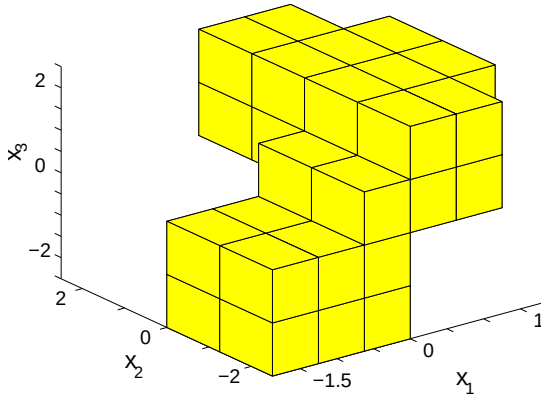
$$f_1(x) = \sum_{j=1}^{n} x_j, \tag{6.4.15}$$

$$f_2(x) = 1 - \prod_{j=1}^{n} (1 - w_j(x_j)), \tag{6.4.16}$$

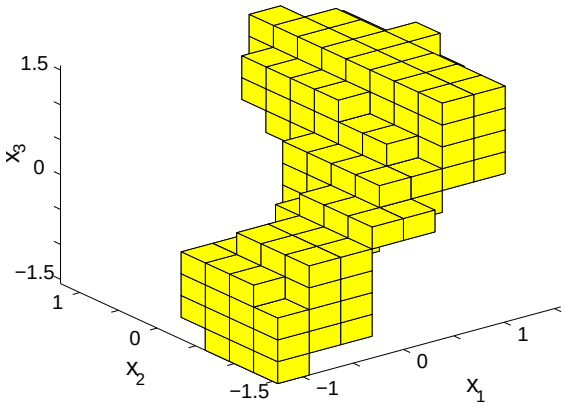
where

$$w_j(z) = \begin{cases} 0.01 \cdot \exp(-(\frac{z}{20})^{2.5}) & \text{for} \quad j = 1, 2 \\ 0.01 \cdot \exp(-\frac{z}{15}) & \text{for} \quad 3 \leq j \leq n \end{cases}$$

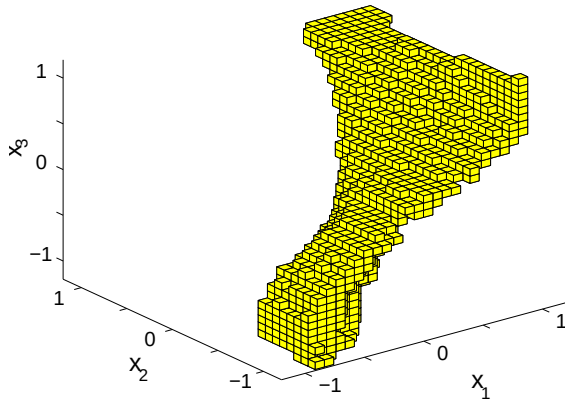
The two objective functions have to be interpreted as follows. f_1 represents the sum of the additional cost necessary for a more reliable production of n items. These



(a) 10 steps



(b) 15 steps



(c) 21 steps

Figure 6.9: Box collections $\mathcal{B}_{10}$, $\mathcal{B}_{15}$ and $\mathcal{B}_{21}$ of MOP (6.4.14).

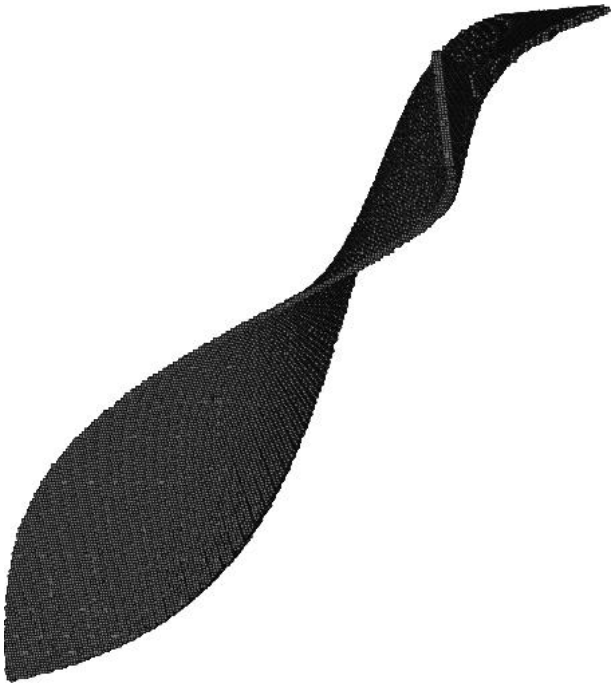


Figure 6.10: The box covering $\mathcal{B}_{33}$ of MOP (6.4.14). The visualization was done by GRAPE (`http://www.iam.uni-bonn.de/sfb256/grape/`)

items are needed for the composition of a certain product. The function f_2 describes the total failure rate for the production of this composed product.

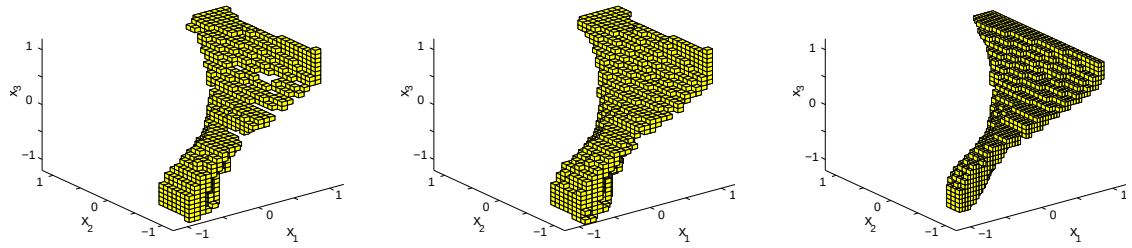
The basic domain is $Q = [0, 40]^n$. For $n = 3$ and $n = 20$ the approximations are shown in Figure 6.12. These were obtained by the sampling algorithm which is capable to detect Pareto optimal solutions on the boundary of the domain Q. In comparison to this we show in Figure 6.6 a covering obtained by the subdivision algorithm on its own combined with a penalization strategy. It can be observed that in this case the use of the sampling algorithm is certainly advantageous.

6.4.5 Example G5 – Optimization of an Active Suspension

In this section we illustrate that the developed algorithms can be useful in applications to real world problems. The following example of optimizing an active suspension is taken from the field of automotive engineering. As it is not the aim of this thesis to focus on suspension technology, the problem will be dealt with only in a reduced form.

It is common in suspension engineering to analyze basic principles of the active suspension regarding system set–up and controller design by using quarter car models. These models consist of the proportional mass of the car body, one wheel and the respective strut[6] ([79], [42]). Due to energy considerations, the active interven-

[6]The link kinematics can be considered by scaling the parameters of the strut by an appropriate translation factor.



(a) Subdivision (b) Recovering (c) Sampling Algorithm

Figure 6.11: Combination of the three algorithms.

tion of the suspension is usually restricted to lower frequencies (e.g. [11]), which are – in good approximation – characterized by the dynamical behavior of the car body only. The basic design of the active suspension can therefore be based on a simple model as illustrated in Figure 6.13.
The road z_0 excites the mass m_B with its coordinate z_B via the strut with spring constant c_{strut} and damping constant d_{strut}. The active suspension allows for an additional active force F_{active}. Using acceleration and level measurements, the car body dynamics can be attuned to the transfer function given in (6.4.17) and described in the Laplace–domain.[7]

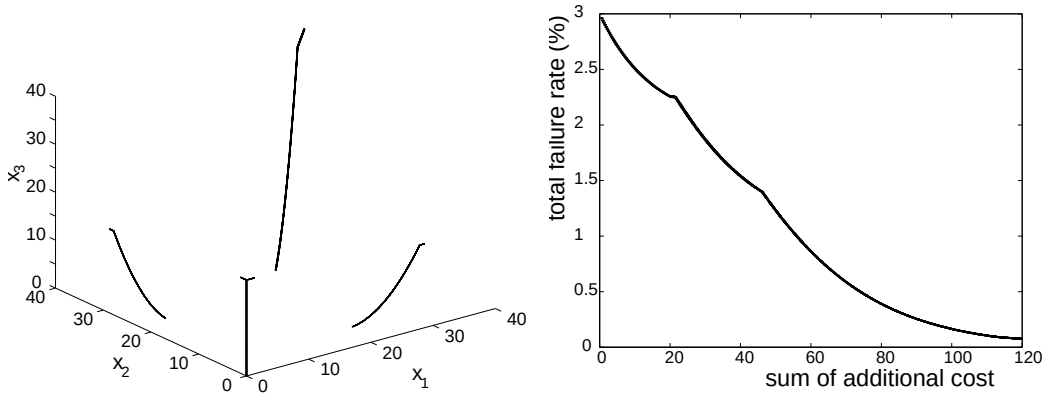
$$G\,(\mathrm{s}) = \frac{z_B}{z_0}\,(\mathrm{s}) = \frac{d\,\mathrm{s} + c}{m_B\,\mathrm{s}^2 + (d + d_s)\,\mathrm{s} + c} \tag{6.4.17}$$

The spring constant c and the damper constant d are composed of the physical constants c_{strut} and d_{strut}, respectively, and an additional controller part using the relative velocity between road and car body. The additional damper term $d_s\,\mathrm{s}$ in the denominator can be obtained by using the absolute car body velocity derived from a measurement of the acceleration.

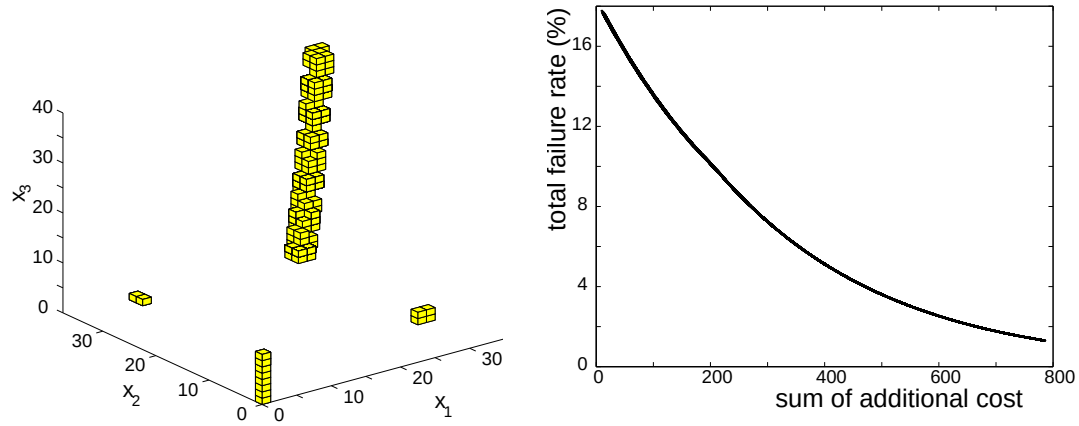
The fundamental design problem of (6.4.17) is depicted in Figure 6.14 (a) (see also [57]). The left–hand side shows the response of the car body to a step on the road of height 1 cm. The spring constant c is kept constant, d and d_s are varied such that $d + d_s = const.$ With increasing d_s, the overshoot decreases, which means that the driving comfort increases.

At the same time it becomes difficult move up a ramp – this fact is illustrated on the right hand side of Figure 6.14 (a). The ramp excitation corresponds to a drive with $20\frac{\mathrm{m}}{\mathrm{sec}}$ on a 15% slope. When the damping of the system is realized by d_s only ($d = 0$), the simulation shows a ramp error of 57 cm, which means that the car hits the mechanical buffers while driving along the slope. Thus, in order to optimize the suspension behavior, it is apparently necessary to consider not only a comfort criterion but also the ramp error.

[7] In order to keep the example simple, actuator influences are disregarded here, as are derivative and measurement filters. An integrating controller part is dispensed with for the same reasons.



(a) Dimension $n = 3$



(b) Dimension $n = 20$

Figure 6.12: Results for MOP (6.4.15): For dimension $n = 3$ after 30 iterations and for dimension $n = 20$ after 100 iterations in parameter space (left) and in image space (right).

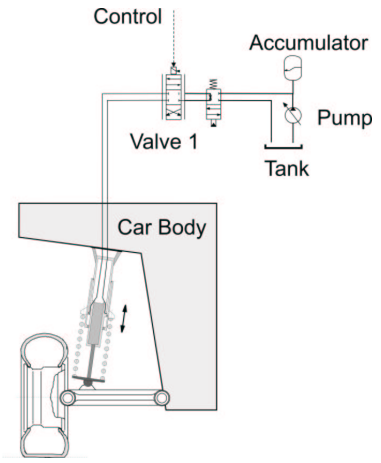


Figure 6.13: Simplified quarter car model

Driving tests show that a body response of 1 Hz and a damping of 0.6 are perceived to be particularly comfortable. This suggests (6.4.18) as a reference car body response with respect to comfort.

$$G_{c.o.}\left(\mathrm{s}\right) = \frac{1}{T_{c.o.}\,\mathrm{s}^2 + 2\,d_{c.o.}\,\mathrm{s} + 1} \qquad | \qquad T_{c.o.} = \frac{1}{2\,\pi}\,\frac{\mathrm{rad}}{\mathrm{sec}}, \quad d_{c.o.} = 0.6 \tag{6.4.18}$$

The difference between (6.4.17) and (6.4.18) can be used as a criterion for comfort. As both functions are minimum phase, it is sufficient to compare the magnitudes of the transfer functions. These considerations lead to (6.4.19) as the first objective function:

$$f_1 = \sum_{j=0,\ldots,450} \left(20\,\log_{10}\left|G\left(i10^{-3+\frac{j}{450}}\right)\right| - 20\,\log_{10}\left|G_{c.o.}\left(i10^{-3+\frac{j}{450}}\right)\right|\right)^2 \tag{6.4.19}$$

As we have already seen in Figure 6.14 (a) it is necessary to add another criterion for the ramp error. Computing the ramp error for $t \to \infty$, we obtain the following second objective function

$$f_2 = \lim_{\mathrm{s}\to 0} \mathrm{s}\,\left(1 - G\left(\mathrm{s}\right)\right)\,\frac{1}{\mathrm{s}^2} = \frac{d_s}{c}. \tag{6.4.20}$$

c, d and d_s are the free parameters for the optimization, m_B is set to 250 kg. All parameters must be positive to avoid rhs–poles and zeros. f_1 and f_2 are both positive penalty functions[8]. Thus

$$f_1, f_2 : \mathbb{R}^{+,3} \to \mathbb{R}^+.$$

The result of the multi–objective optimization is shown in Figure 6.14 (b).

[8]For numerical reasons, the parameters lie within the intervals

$$c \in [1, 20000] \qquad d \in [0, 5000] \qquad d_s \in [0, 5000]$$

As the result shows, these restrictions have no impact on the optimization.

The Pareto set can be divided into two parts: the upper part in parameter space with $d_s > 0$ belongs to the right hand part of the 'Objectives'–diagram in Figure 6.14 (b). This part is magnified in the inner figure. The second part in parameter space lies in the plane where $d_s = 0$. The corresponding curve in image space has an extremely steep gradient. However, from a physical point of view this branch is irrelevant. The extremal substationary points of the multi–objective optimization problem are therefore the points 1 and 2 (see left hand side in Figure 6.14 (b)). Point 1 is ramp–oriented, point 2 comfort–oriented.

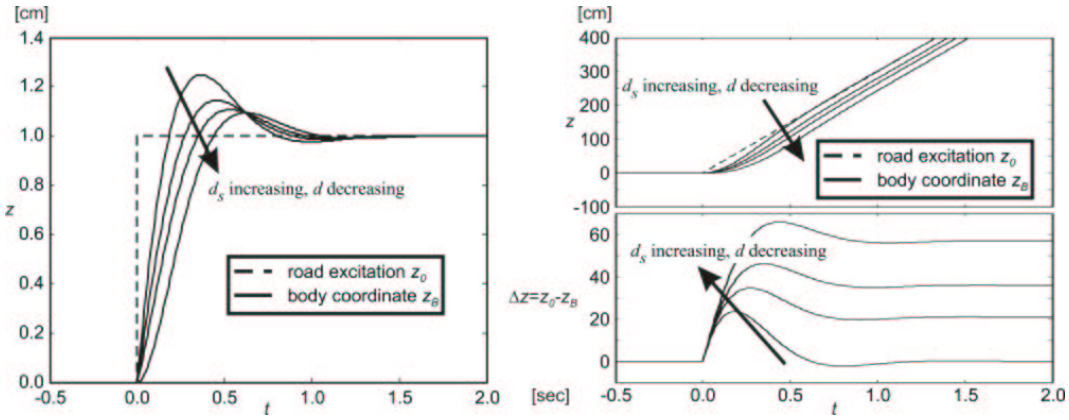
In Figure 6.14 (c) we show the time and frequency response of the car body corresponding to both of these points. The comfort–oriented substationary point yields the prescribed comfort optimum (6.4.18), which can be seen both in the frequency and in the time domain. The ramp–oriented substationary point returns a ramp error of 0 (only suggested by Δz_{ramp} in Figure 6.14 (c), but it can clearly be seen in Figure 6.14 (b)). However, with 0.26, the damping of this system is unnecessarily low[9]. This is due to the choice of the comfort criterion f_1. The ramp–oriented frequency response in Figure 6.14 (c) illustrates the reason: its deviation from the comfort optimum around its peak response is weighed in the same way as the deviation at higher frequencies far below 0 dB gain. An increase of damping using d instead of d_s (and thus improving damping without deteriorating the ramp error) leads to a lower peak–response but also to higher gain at higher frequencies. This problem could be dealt with by the use of penalty functions. However, further improvements of the objective functions will be discussed elsewhere.

The optimized set of parameter values determined in the way presented here allows for a controller tuning according to the target customer group. The suspension design process can be simplified, as test drivers just need to switch between Pareto points and thus do not have to tune all free parameters without additional support. In addition, multi–objective optimization of suspension controllers also has the potential for further suspension improvements. For instance, in [57] it is shown that the Pareto set can be used for the *self–optimization* of a car.

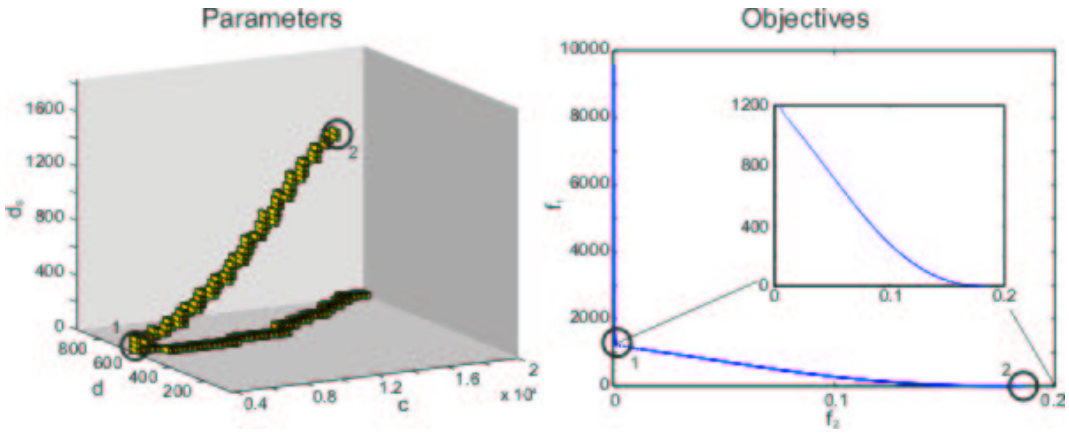
6.5 A Data Structure for the Computation of the Nondominance Problem

In this section we propose a data structure for the efficient computation of the nondominance problem which occurs in most multi–objective optimization algorithms – such as in the Sampling Algorithm or in the algorithm ND-Cont which is described in Section 6.8.
After a detailed description of the method we illustrated its strength by a comparison both with the linear list approach and the quad tree approach on a category of problems. The computational results indicate that the method is particularly advantageous in the case where the proportion of the nondominated vectors versus the total set of criterion vectors is not too large. The data structure was developed

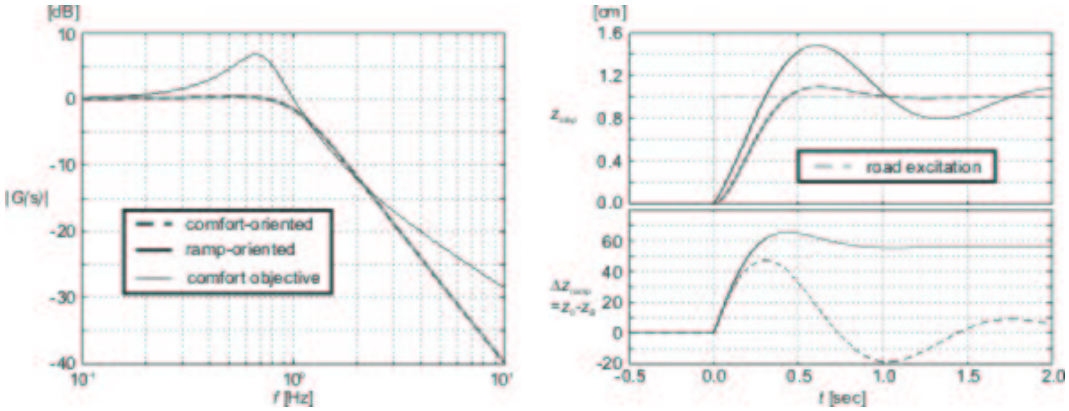
[9] But it still lies within the range of today's suspensions.



(a) Step and ramp response of the car body



(b) Result of the multi–objective optimization using the sampling algorithm



(c) Time and frequency responses of the extreme Pareto points

Figure 6.14: Optimization of an active car suspension

by the author in [107].

6.5.1 Introduction and Background

In most computational algorithms for the solution of a multi–objective optimization problem

$$\min F : Q \subset \mathbb{R}^n \to \mathbb{R}^k \tag{6.5.21}$$

the problem arises to sort out the nondominated vectors from a given finite (but large) set of criterion vectors $P \subset \mathbb{R}^k$. A vector v is called nondominated in P if there is no vector $p \in P$ which dominates v.
The *nondominance problem* can be divided into two main classes. First, there is the *static* nondominance problem. Here one has to find the subset N of nondominated vectors of a given set P at once. For details we refer e.g. to [95] and [51], where the problem is solved up to $k = 4$.
Second, there is the *dynamic* nondominance problem which occurs in most multi–objective optimization techniques and which we want to address in this section. We are given a set of nondominated vectors P, and, in addition to this set, there is a sequence of candidates (which is generated by the optimization procedure, e.g. the Sampling Algorithm described in Section 6.3). For every vector v of this sequence the *archive* P has to be updated (see Figure 6.15).

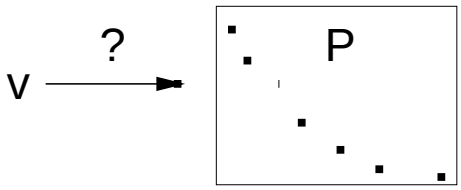


Figure 6.15: Scheme of the dynamic nondominance problem: a given archive P of nondominated points has to be updated by arriving data.

There are several alternative approaches for the solution of this problem. First, if all the candidates (and hence all elements of the archive P) lie in bounded (hyper–) rectangles, it is suitable to use *kd–trees* ([5, 19]) or *range trees* ([6, 19]). *Priority trees* ([77]) are suitable for the case where these rectangles are unbounded on a single side. If there are no restrictions to the range of the objectives the intuitive *linear list* approach (see e.g. [117]) can be used. Another way of attacking the problem is proposed in [52], where a clever usage of the data structure *quad tree* (see [41]) is utilized. These techniques were refined in [117] and [116]. We refer to [84] for the extensions of the quad tree approach to multi–objective evolutionary algorithms. Furthermore, there exists the *composite point* approach which is presented in [39]. The data structure we are proposing here is – like all the methods mentioned above except the linear list approach – tree–based.

6.5.2 Attacking the Nondominance Problem

Let us assume that we have a dynamic nondominance problem, i.e. a sequence of candidates $v^j \in \mathbb{R}^k$ for which a given archive $P \subset \mathbb{R}^k$ has to be updated.
The basis for our approach is to store the nondominated vectors from P in the following tree:

DEFINITION A k-ary tree T is called a *dominance decision tree*, if for every node $p = (p_1, \ldots, p_k) \in T$ and for each existing i-th son $s = (s_1, \ldots, s_k)$ from p the following holds:

$$s_j \leq p_j \quad \forall j = 1, .., i-1 \tag{6.5.22}$$

$$s_i > p_i \tag{6.5.23}$$

A simple example of a dominance decision tree for three objectives is shown in Figure 6.16.

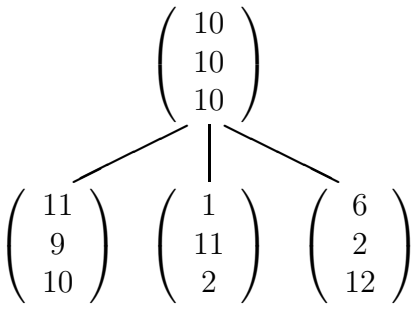


Figure 6.16: Example of a dominance decision tree for $k = 3$.

For a given archive P and a new candidate $v \in \mathbb{R}^k$ the following steps have to be performed:

1.) If there exists one vector $D \in P$ which dominates v, then STOP, else go to step 2.

2.) Detect and delete all elements $d \in P$ which are dominated by v.

3.) Insert v into the archive P.

In the following we describe how to realize these steps and how to take advantage of the structure of the dominance decision tree.
ad 1.) Assume that a vector v and an archive P – stored in a dominance decision tree with root r – are given. First, v has to be compared to the root r. If r dominates v, we stop and v has to be discarded. If v and r are mutually non–dominating, then the algorithm has to make the comparisons recursively in some subtrees of r. Due to (6.5.23) this comparison has to be made only in the i-th subtrees of r where $r_i \leq v_i$. The algorithm `DetectDomination` reads as follows:

Algorithm DetectDomination

Input: root r, vector $v \in \mathbb{R}^k$.

Task: returns 1, if there exists a vector $p \in P$ (given by root r) which dominates v, else 0.

```
DetectDomination (root r,node v)
   if r dominates v
      return 1
   for i = 1,...,k
      if r_i ≤ v_i AND the i-th son of r exists (denote it by r → son_i)
         if DetectDomination(r → son_i, v) == 1
            return 1
   return 0
```

ad 2.) Assume again that a vector v and a dominance decision tree P are given. First we have to discuss which nodes have to be checked for domination, i.e. in which subtrees of P the algorithm has to look for dominated points. With given $p \in P$ it follows by conditions (6.5.22) and (6.5.23) that the search has to be continued in the first i subtrees of k where $i \in \{1, \ldots, k\}$ is the smallest index where $v_i > p_i$. In order to see this let s be a vector from the l-th subtree where $i < l \leq k$. Then by construction of the dominance decision tree:

$$s_i \overset{(6.5.22)}{\leq} p_i < v_i,$$

and hence s cannot dominate v.
We illustrate this by an example: let p, v_1, v_2 and v_3 be given by

$$p = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix},\ v_1 = \begin{pmatrix} 12 \\ 8 \\ 5 \end{pmatrix},\ v_2 = \begin{pmatrix} 2 \\ 12 \\ 1 \end{pmatrix},\ v_3 = \begin{pmatrix} 3 \\ 3 \\ 3 \end{pmatrix}.$$

In case of $v = v_1$ the search has only to be continued in the first subtree of v_1, whereas with the choice of $v = v_2$ the algorithm has to search in the first two subtrees of v_2. Eventually in case of $v = v_3$ the data structure has no advantage because all three subtrees have to be scanned.
A deletion of a node $p \in P$ can be done as follows: if one son s of p does exist, then it can be moved to the position of p. The other nodes of the subtree of p have to be reinserted – into the lifted subtree with root s. The deletion of the dominated nodes can be done via one postorder run through the tree:

Algorithm DeleteDominated

Input: root r, vector $v \in \mathbb{R}^k$.

Task: deletes every vector $p \in P$ which is dominated by v.

```
DeleteDominated (root r, vector v)
   for i = 1, ..., k
      if the i-th son of r exists (denote it by r → son_i)
         DeleteDominated (r → son_i, v)
      if v_i > r_i
         break
   if v dominates r
      if r is leave
         delete r and STOP
      j := arg min { the j-th son of r exists (denote it by r → son_j) }
      Move r → son_j to the position of r
      for l = j + 1, ..., k
         if the l-th son of r exists (denote it by r → son_l)
            TreeInsert (r → son_l)
      delete r
```

The algorithm `TreeInsert` used above reads as follows:

Algorithm TreeInsert

Input: root r, root s.

Task: inserts every vector of the tree given by root s into the tree given by root r.

```
TreeInsert (root r, root s)
   for i = 1, ..., k
      if the i-th son of s exists (denote it by s → son_i)
         TreeInsert (r, s → son_i)
   Insert (r, s)
   delete s
```

ad 3.) By its definition there is only one possible way for the insertion of a vector v into a given dominance decision tree P (given by root r):

Algorithm Insert

Input: root r, vector v.

Task: inserts v into the archive P (given by root r).

```
Insert(root r, vector v)

   i := arg min {v_i > r_i}
   if the i-th son of r exists (denote it by r → son_i)
      Insert(r → son_i,v)
```

else
$r \rightarrow \mathsf{son}_i := v$

Now the main algorithm for the update of an archive P (with root r) by a candidate v can be stated. Note that the root of the dominance decision tree can be changed in the algorithm `DeleteDominated`.

Algorithm Update

Input: root r, vector v.

Task: updates the archive P (given by root r) by the candidate v.

```
Update (root r, vector v)
  if( P is empty)
    P := {v}   (r := v)
    STOP
  if DetectDomination (r, v) == 1
    STOP
  DeleteDominated (r, v)
  Insert ((root of the archive P), v);
```

The algorithm presented above has the average case complexity of $O(n^2)$ for vector comparisons ([36]). However, since there is no algorithm with a provable complexity better than $O(n^2)$, we will discuss the particular advantages of the dominance decision tree approach in the following section.

6.5.3 Computational Results

Here we make a comparison of the approaches which need no restrictions to the range of the objective values. We compare the linear list approach, the quad tree approach and the dominance decision tree approach.
For the comparison we proceed as in [117] and take test points generated by an annulus as criterion vectors because by this category of problems the particular advantages of the three approaches can be demonstrated.
We choose a sequence of vectors $v^j \in \mathbb{R}^k$ which have to be inserted to the archive P given by the nondominated vectors of the set $\{v^1, .., v^{j-1}\}$. The components of every vector $v^j = (v_1, .., v_k)$ are of the following form:

$$v_i := -\frac{\tilde{r}_i}{\|w\|} w_i, \tag{6.5.24}$$

where $\tilde{r}_i \in [r, 1]$ and $w \in \mathbb{R}^k_+$ are chosen at random. This choice of criterion vectors allows to adjust not only the number k of "objectives" but also the proportion p_n of the nondominated vectors versus the total number of criterion vectors: it is easy to see that the larger the value of $r \in [0, 1)$ is the larger the value of p_n will typically

be. Exactly this proportion is important for the comparison of the two tree based approaches: the computational results indicate that the dominance decision tree approach is advantageous in the case where the proportion p_n is "moderate". The larger the value of p_n the better is the performance of the quad tree approach and it gets eventually faster than the dominance decision tree approach. Of course it is barely possible to detect an exact "balance proportion" p_n^b where the two approaches have the same running time, but at least it seems to be possible to give some guidelines.
In Figure 6.17 and Table 6.1 we show that for $k = 3$ the proportion where the running time of the two approaches is basically the same is approximately $p_n^b = 1/3$ (with $r_b = 0.95$). That means that the dominance decision tree approach is faster when the optimization algorithm generates in average at most every third time a nondominated vector, otherwise the quad tree approach is faster.
For $k = 4$ the proportion p_n^b seems to be 0.5 (see Figure 6.18 and Table 6.2), but because of the higher dimension the limit radius $r_b \approx 0.85$ is lower than for k equals 3. A similar observation was made for $k = 5$ ($p_n^b \approx 0.5$ but $r_b \approx 0.7$). This means that the efficiency of the quad tree approach increases with growing k in comparison to the dominance decision tree approach.
In the case where k equals 2 we figured out that the linear list approach is faster than the dominance decision tree approach as well as the quad tree approach. This is possibly due to the overhead given by the tree based approaches.

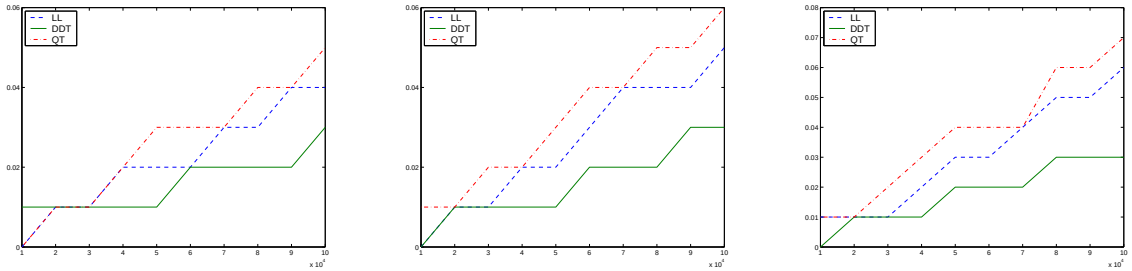
6.6 Extensions for Non–Smooth Models

6.6.1 Introduction

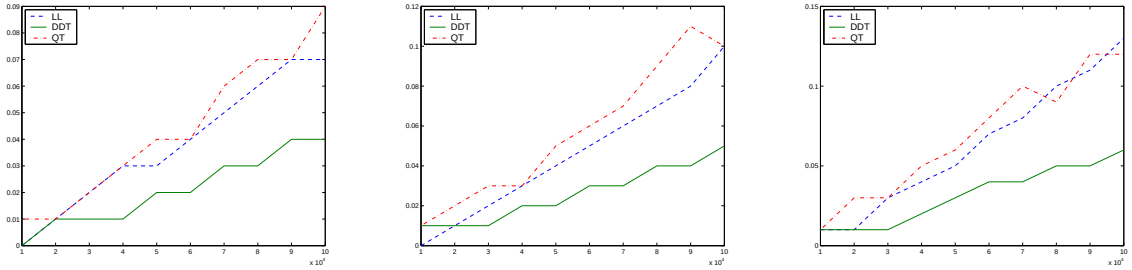
This section introduces some modifications on the basic algorithms (see Section 6.3) which are advantageous in case the derivatives of the objectives of the MOP are not available. The state of the work presented here is under investigation and the algorithms have to be refined in the future. The content of this section is partly developed by the author in [108].

6.6.2 A Short Introduction to MOEA's

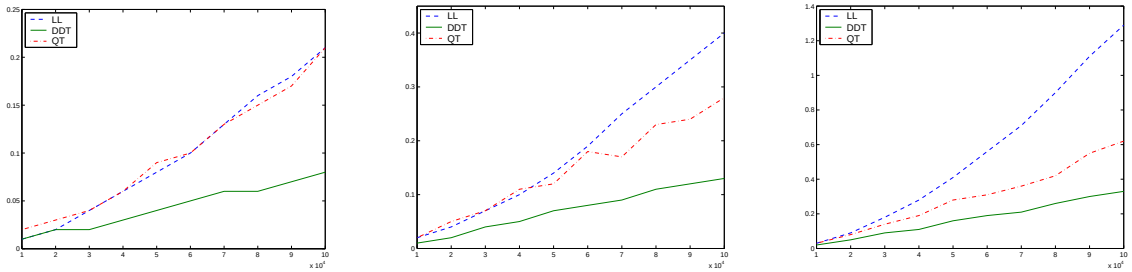
Evolutionary algorithms (EAs) are iterative stochastic search methods that are based on the two concepts of generate and evaluate [16]. Up to now, there are many **M**ulti–objective **O**ptimization methods that are based on this idea of **EA**s (MOEAs). MOEAs have demonstrated the advantage of using population–based search algorithms for solving multi–objective optimization problems. In all of these methods converging to the Pareto–optimal front and maintaining a spread of solutions (diversity) are the most important factors.
MOEAs can be divided into two groups. The first group contains the MOEAs that always keep the best solutions of each generation in an *archive*, and they are called MOEAs with elitism. It is proved by Rudolph ([101, 100, 102]) that in some cases elitism will provide converging to the true Pareto–optimal front.



(a) r=0.1 (b) r=0.2 (c) r=0.3



(d) r=0.4 (e) r=0.5 (f) r=0.6



(g) r=0.7 (h) r=0.8 (i) r=0.9

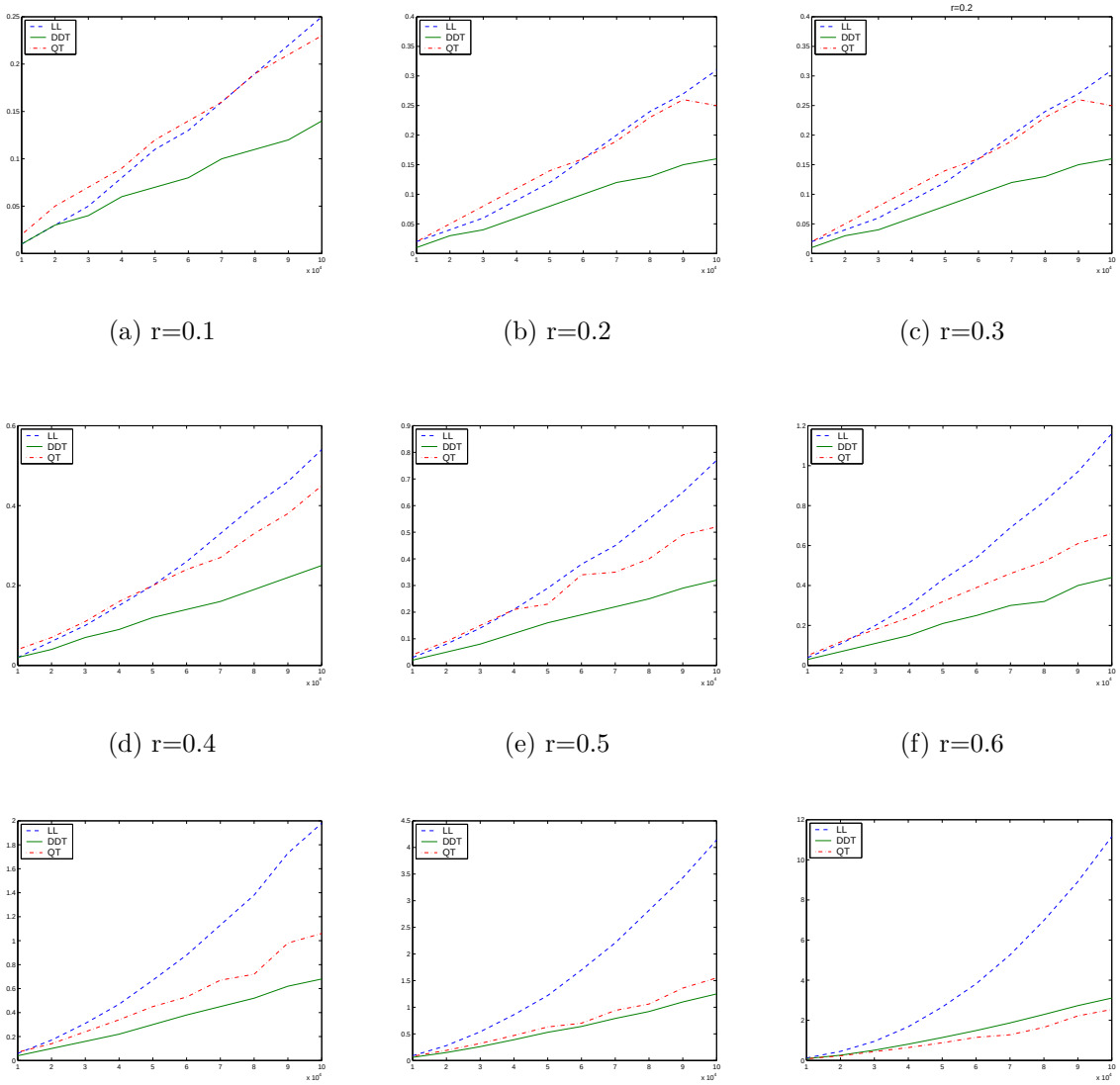
Figure 6.17: Annulus generated test problem results for $k = 3$. In the Figures the number N of criterion points versus the running time of the three approaches is plotted for different values of the radius r. Here we have chosen $N = \{1000, 2000, \ldots, 10000\}$. For details see Table 6.1.

Table 6.1: Annulus generated test problem results for $k = 3$

	N	2000	4000	6000	8000	10000
r = 0.3	T_{LL}	0.01	0.02	0.03	0.05	0.06
	T_{QT}	0.01	0.03	0.04	0.06	0.07
	T_{DDT}	0.01	0.01	0.02	0.03	0.03
	p_n	0.07	0.04	0.04	0.03	0.03
r = 0.5	T_{LL}	0.01	0.03	0.05	0.07	0.10
	T_{QT}	0.02	0.03	0.06	0.09	0.10
	T_{DDT}	0.01	0.02	0.03	0.04	0.05
	p_n	0.10	0.07	0.06	0.05	0.04
r = 0.7	T_{LL}	0.02	0.06	0.10	0.16	0.21
	T_{QT}	0.03	0.06	0.10	0.15	0.21
	T_{DDT}	0.02	0.03	0.05	0.06	0.08
	p_n	0.15	0.11	0.09	0.07	0.07
r = 0.9	T_{LL}	0.09	0.28	0.56	0.90	1.29
	T_{QT}	0.08	0.19	0.31	0.42	0.62
	T_{DDT}	0.05	0.11	0.19	0.26	0.33
	p_n	0.37	0.28	0.23	0.20	0.18
$r_b = 0.95$	T_{LL}	0.29	0.85	1.66	2.66	3.91
	T_{QT}	0.13	0.41	0.69	1.05	1.33
	T_{DDT}	0.15	0.40	0.65	0.95	1.31
	p_n	0.61	0.47	0.41	0.36	0.33

Table 6.2: Annulus generated test problem results for $k = 4$.

	N	2000	4000	6000	8000	10000
r = 0.3	T_{LL}	0.05	0.11	0.21	0.30	0.39
	T_{QT}	0.06	0.12	0.19	0.28	0.38
	T_{DDT}	0.03	0.07	0.11	0.16	0.20
	p_n	0.18	0.14	0.12	0.10	0.10
r = 0.5	T_{LL}	0.08	0.21	0.38	0.55	0.77
	T_{QT}	0.09	0.21	0.34	0.40	0.52
	T_{DDT}	0.05	0.12	0.19	0.25	0.32
	p_n	0.26	0.20	0.17	0.15	0.14
r = 0.7	T_{LL}	0.17	0.47	0.88	1.38	1.98
	T_{QT}	0.14	0.34	0.53	0.72	1.06
	T_{DDT}	0.10	0.22	0.38	0.52	0.68
	p_n	0.42	0.32	0.28	0.25	0.22
$r_b = 0.85$	T_{LL}	0.20	0.71	1.80	3.36	5.25
	T_{QT}	0.13	0.29	0.56	0.76	1.20
	T_{DDT}	0.11	0.30	0.51	0.74	1.05
	p_n	0.68	0.56	0.50	0.46	0.43
r = 0.9	T_{LL}	0.45	1.68	3.82	6.98	11.16
	T_{QT}	0.23	0.64	1.15	1.65	2.54
	T_{DDT}	0.26	0.81	1.49	2.29	3.11
	p_n	0.85	0.77	0.71	0.66	0.63



(a) r=0.1 (b) r=0.2 (c) r=0.3

(d) r=0.4 (e) r=0.5 (f) r=0.6

(g) r=0.7 (h) r=0.8 (i) r=0.9

Figure 6.18: Annulus generated test problem results for $k = 4$. For details see Figure 6.17 and Table 6.2.

In the second group, there is no archive for keeping best solutions and MOEA may lose them during generations. MOEAs with elitism are studied in several methods like Rudolph's Elitist MOEA, Elitist NSGA–II, SPEA, PAES (see [21] for all) and SPEA2 [125].

Figure 6.19 shows the typical structure of a MOEA with elitism, where t denotes the number of the generation, P_t the population, and A_t the archive at generation t. The aim of function *Generate* is to generate new solutions in each iteration t which

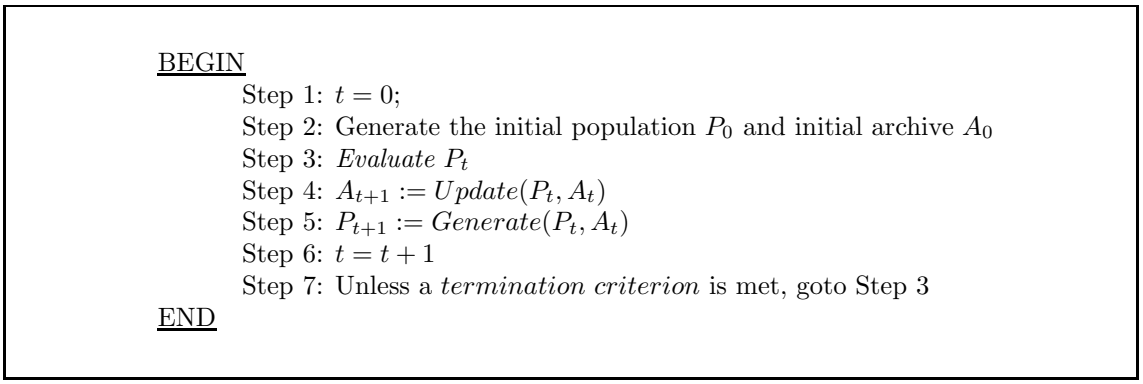


Figure 6.19: Typical structure of an archive–based MOEA.

is done through selection, recombination and mutation. The function *Evaluate* calculates the *fitness* value of each individual in the actual population P_t. Fitness assignment in MOEA is done in different ways such as by Pareto–ranking [47], non–dominated sorting [22], or by calculating Pareto–strengths [126]. Since only the superior solutions must be kept in the archive, it must be updated after each generation. The function *Update* compares whether members of the current population P_t are non–dominated with respect to the members of the actual archive A_t and how and which of such candidates should be considered for insertion into the archive and which should be removed. Thereby, an archive is called *domination–free* if no two points in the archive do dominate each other. Obviously, during execution of the function *Update*, dominated points must be deleted in order to keep the archive domination–free.

These three phases of an elitist MOEA are iteratively repeated until a termination criterion is met such as a maximum number of generations or when there has been no change in non–dominated solutions found for a given number of generations. The output of an elitist MOEA is the set of non–dominated solutions stored in the final archive. This set is an approximation of the Pareto–set and often called *quality set*.

The above algorithm structure is common to most elitist MOEAs. In some of these methods (e.g. Rudolph and Agapie' Elitist GA, NSGA2, ...), in the case of inadequate available space in the archive to store all of the non–dominated solutions, only those non–dominated solutions that are maximally apart from their neighbors are chosen. Therefore a crowding method is done to select the solutions in less crowded areas. However, the true convergence property cannot be achieved, since an existent Pareto–optimal solution may get replaced by a non–Pareto–optimal during

the crowding selection operation. In some other methods (e.g., SPEA) when the size of the archive exceeds, clustering method is done among the archive members. The use of clustering among the archive members guarantees spread among them. However, these algorithms lack a convergence proof, simply because of the same reason as in crowding methods, during the clustering procedure an existent Pareto–optimal archive member may get replaced by a non–Pareto–optimal.

Discussion: As it is explained, the elitist MOEAs are suitable candidates to find Pareto–fronts for different kinds of multi–objective problems ([126], [125], [22], [102], $\cdots$). But we have to consider that convergence and diversity are not satisfied by these methods when solving some test functions. For example in the explained methods, updating the archive causes difficulties in convergence. On the other hand, keeping a good spread of solutions in the Pareto–front, for gaining more diversity emphasizes the regions in the front that are less crowded, which may cause to loose the better convergence.
Anyway, having a good spread of solutions in the Pareto–front together with a good convergence is always desired and makes the investigators to compare the performance of their methods according to these metrics [71].

6.6.3 The Algorithms

Basic idea The algorithms proposed in this section are all based on the following observation:

MOEAs (typically) generate very quickly some
very good approximations of Pareto points.

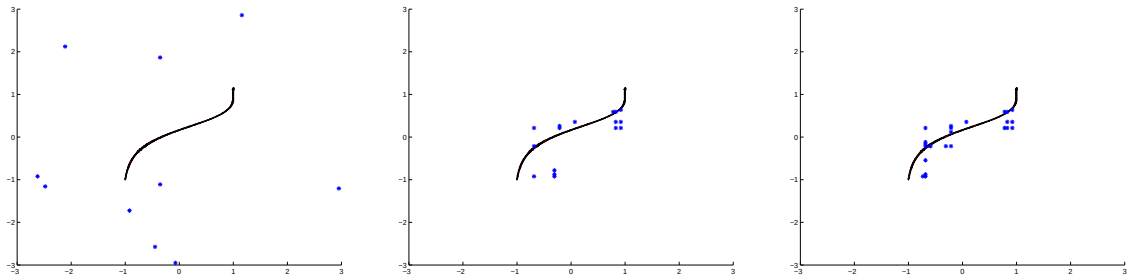
Let this be indicated by an example. Figure 6.20 shows the Pareto set of MOP (6.4.12) which is also used for the illustration of the algorithms proposed in this section:

$$\begin{aligned} f_1, f_2 &: Q \subset \mathbb{R}^2 \to \mathbb{R} \\ f_1(x) &= (x_1 - 1)^2 + (x_2 - 1)^4 \\ f_2(x) &= (x_1 + 1)^2 + (x_2 + 1)^2 \end{aligned} \tag{6.6.25}$$

Figure 6.20 (a) shows a start population consisting of 10 randomly chosen points in the domain $Q = [-3, 3] \times [-3, 3]$. The following two figures show the resulting populations after 5 and 10 generations using SPEA. It is evident that even after only 5 generations there are some individuals close to the Pareto set.

This property makes it possible to improve the sampling algorithm described above: instead of using many test points to evaluate a (high–dimensional) box, it is better to take just a few test points as the initial population of a "short" MOEA[10].

[10] A short MOEA is characterized by a short running time; that means small initial population and few generations



(a) initial population (b) 5 generations (c) 10 generations

Figure 6.20: One advantage of EAs is to find some good solutions quickly. The solid line indicates the actual Pareto set.

The EA only has to run for a short time because a box is kept if it contains at least only one "good" point (in this case a nondominated point).

The Algorithms

EA–subdivision The discussion made above leads directly to the first algorithm: use the sampling algorithm combined with a "short" MOEA for the evaluation of every box. The only modification to the sampling algorithm described in Section 6.3.3 is given by:

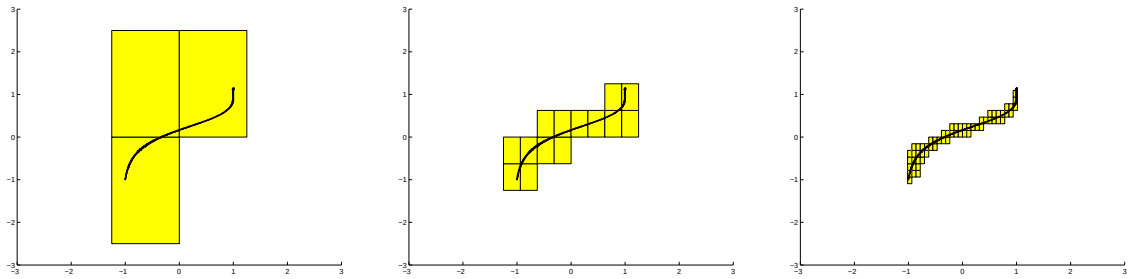
$$P_B := \text{final population of "short" MOEA}$$

The only task of the MOEA is to find as fast as possible one good approximation of a Pareto point relative to the given domain. So here no diversity or even clustering are needed. But special attention should be paid so that the MOEA does not get stuck on local minima. "Hill climbers" have not been tested successfully in some cases.

EXAMPLE 6.6.1 Figure 6.21 shows the coverings of the set of Pareto points after 4, 8 and 12 subdivision steps. The black "line" indicating the Pareto set is in fact the resulting box collection after 20 subdivision steps (compare to Example G1). In this example the population size and the number of generations were chosen as 5.

Recovering As in most of the subdivision algorithms, in the EA–subdivison algorithm the problem remains that boxes which contain a part of the Pareto set can be sorted out in the selection step, e.g. when the MOEA is too short. As a kind of "healing" process we describe two algorithms in this section to recover the Pareto set.

Let us first consider the case where the covering is not complete but every box contains a part of the Pareto set (like box B_1 in Figure 6.22). The aim of the algorithm is to extend the given box collection step by step along the covered parts



(a) 4 steps (b) 8 steps (c) 12 steps

Figure 6.21: Application of EA–subdivision

of the Pareto set until no more boxes are added. In order to find the corresponding neighboring boxes of a given box B with center c and radius r we run a MOEA in the extended box $\hat{B}$ given by center c and radius $\lambda \cdot r$ with $\lambda > 1$, say $\lambda = 3$. Afterwards the box collection is extended by the boxes $B \in \mathcal{P}_k$ which contain points from the resulting population (see Figure 6.23). In the first step this is done for all boxes from the box collection, for the following steps this local search has to be done only in the neighborhood of the boxes which were added in the preceeding step.

With a given box collection $\mathcal{B}_k$ the complete algorithm `StaticRecover` reads as follows:

Algorithm `StaticRecover`

1.) for all $B \in \mathcal{B}_k$
    $B.active := TRUE$
2.) for $i = 1, .., MaxStep$
    $\hat{\mathcal{B}}_k := \mathcal{B}_k$
    for all $B \in \mathcal{B}_k$: $B.active == TRUE$
        compute MOEA in extended universe $\hat{B} := (B.c, \lambda \cdot B.r)$
        $P :=$ final population
        $B.active = FALSE$
        for all $p \in P$:
            if $B(p,k) \notin \mathcal{B}_k$
                $\mathcal{B}_k := \mathcal{B}_k \cup B(p,k)$
                $B(p,k).active = TRUE$
    if $\hat{\mathcal{B}}_k == \mathcal{B}_k$ $STOP$

Hence `StaticRecover` only allows the addition of boxes into the given collection. The desired covering of the set of Pareto points cannot get worse, but will improve if the parameters of the algorithm are adjusted properly. On the other hand, `StaticRecover` does not treat adequately the case where a box does not contain some part of the Pareto set but is possibly far away (e.g. box B_2 in Figure

6.22). In this case the algorithm would extend the box covering by many undesired regions on their way towards the Pareto set (in particular in higher dimensions). Thus, when there are "good" and "bad" boxes like in Figure 6.22 we propose the application of the following algorithm.

Algorithm `DynamicRecover`

1.) for all $B \in \mathcal{B}_k$
    $B.active := TRUE$
2.) for $i = 1, .., MaxStep$
    $\hat{\mathcal{B}}_k := \mathcal{B}_k, \quad \mathcal{B}_k := \emptyset$
    for all $B \in \hat{\mathcal{B}}_k$: $\quad B.active == TRUE$
        compute MOEA in extended universe $\hat{B} := (B.c, \lambda \cdot B.r)$
        $P_B :=$ final population
    $P :=$ nondominated points of $\bigcup_{B \in \hat{\mathcal{B}}_k} P_B$
    for all $p \in P$:
        $\mathcal{B}_k := \mathcal{B}_k \cup B(p,k)$
        if $B(p,k) \in \mathcal{B}_k \quad B(p,k).active := FALSE$
        else $\qquad\qquad B(p,k).active := TRUE$
    if $\hat{\mathcal{B}}_k == \mathcal{B}_k \qquad STOP$

In contrast to `StaticRecover` this algorithm has again the disadvantage that good boxes can be deleted while they have been computed once. This will be addressed in the next subsection. The speed of the algorithm depends – besides of the MOEA – on the choice of the extension factor λ. A larger value of λ yields faster convergence but lower robustness. In general, the number of generations and the size of the initial population should increase with λ. For this local covering of the part of the Pareto set the MOEA has to preserve diversity. Furthermore the convergence of the MOEA should be good enough in order not to insert too many superfluous boxes.

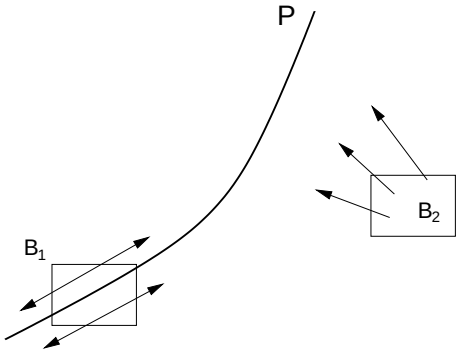
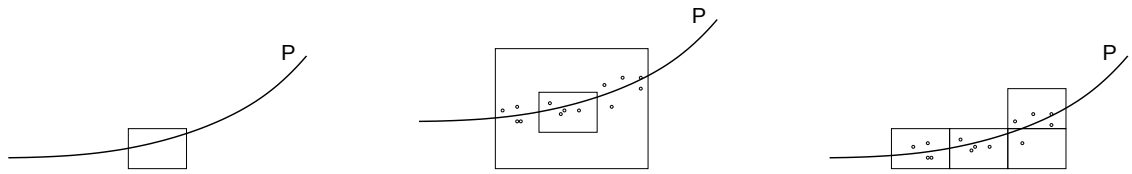


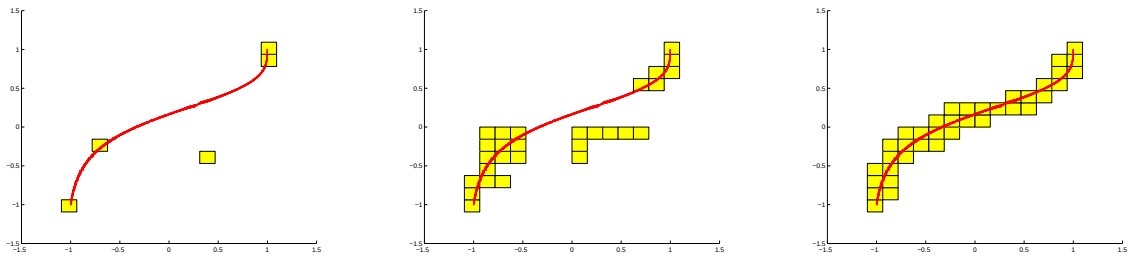
Figure 6.22: Different problems for recovering



(a) covering (b) local search (c) recovering

Figure 6.23: Working principle of StaticRecover

EXAMPLE 6.6.2 Here again the MOP (6.6.25) is considered. The algorithm DynamicRecover was applied to a chosen initial box collection (see Figure 6.24). The algorithm stops after two iterations with a total covering of the Pareto set.



(a) given "covering" (b) recover step 1 (c) recover step 2

Figure 6.24: Application of DynamicRecover on MOP (6.6.25).

6.6.4 Using Archives

So far in this chapter we have developed algorithms for the *covering* of Pareto sets. This works particularly well in case the objectives are smooth and/or the dimension of the MOP is moderate: in a (not infinitesimal small) neighborhood of an optimal solution there are typically other points which have similar properties, in particular in the context of multi–objective optimization. Furthermore, having computed a tight covering, the set of interest which is contained in the boxes can be retrieved at any time with negligible computational effort.
This does not hold when the objectives are "rough". In this case it can happen more often that boxes which contain optimal solutions and which were even selected once to be relevant can get lost in the course of the computation. Thus, for the numerical treatment of these problems it seems advisable to capture the best solutions which are detected so far, i.e. to store the nondominated points in an archive.
In the following we give an extension of the algorithm `DynamicRecover` which carries out this idea. Roughly speaking, the algorithm `DynamicRecover with Archiving`

(DRA) performs DynamicRecover on the box collection $\mathcal{B}_i$ resulting from the current archive A_i, which is itself permanently updated by the result of the local search of DynamicRecover.

Given an initial set of (nondominated) points A_0, $R_0 := A_0$, and an insertion depth d the algorithm DRA reads as follows:

Algorithm DynamicRecover with Archiving

for $i = 0, 1, 2, \ldots$

(i) $\mathcal{B}_i := \bigcup_{a \in A_i} B(a, d)$

(ii) for all $B \in \mathcal{B}_i$: $\exists a \in B \cap A_i \cap R_{i-1}$ $\qquad (*)$
compute MOEA in extended universe $\hat{B} := (B.c, \lambda \cdot B.r)$
$P_B :=$ final population
$R_i :=$ nondominated points of $\bigcup_{B \in \mathcal{B}_i} P_B$

(iii) $A_{i+1} :=$ nondominated points of $A_i \cup R_i$

If in addition to the procedure described above a global optimization algorithm (e.g. a MOEA with elitism) is run on the archive, it is possible to prove convergence to the Pareto set[11]. To do this, we now formulate the (basic) algorithm DRA II.

Algorithm DRA II

$P_0 \subset S_0$ drawn at random
$A_0 :=$ nondominated points of P_0

for $j = 0, 1, 2, \ldots$
$P_{j+1} :=$ generate (P_j)
$R_{j+1} :=$ DynamicRecover (A_j)
$A_{j+1} :=$ nondominated points of $A_j \cup P_{j+1} \cup R_{j+1}$

For convenience of the reader, we recall in the following three required definitions before we state some convergence results.

Definition 6.6.3 Let a MOP $F : \mathbb{R}^n \to \mathbb{R}^k$ be given. A point $x \in \mathbb{R}^n$ is *weakly Pareto optimal* if there does not exist another point $y \in \mathbb{R}^n$ such that $F(y) <_p F(x)$.

Definition 6.6.4 Let $u \in \mathbb{R}^n$ and $A, B \subset \mathbb{R}^n$. The semi-distance $\mathrm{dist}(\cdot, \cdot)$ and the *Hausdorff distance* $d(\cdot, \cdot)$ are defined as follows:

(a) $\mathrm{dist}(u, A) := \inf_{v \in A} \|u - v\|$

(b) $\mathrm{dist}(B, A) := \sup_{u \in B} \mathrm{dist}(u, A)$

[11] In case the state space is discrete, convergence results can be found [100] and [102].

(c) $d(A,B) := \max\{\text{dist}(A,B), \text{dist}(B,A)\}$

DEFINITION 6.6.5 Let $X, X_1, X_2, \ldots$ be random variables on a probability space (Ω, Σ, μ). If

$$\lim_{n\to\infty} X_n(\omega) = X(\omega)$$

for μ-almost all $\omega \in \Omega$, we say that

$$\lim_{n\to\infty} X_n = X \quad \text{with probability one.}$$

THEOREM 6.6.6 *Let an MOP $F : Q \subset \mathbb{R}^n \to \mathbb{R}^k$ be given, where $Q = [a_1, b_1] \times \ldots \times [a_n, b_n] \subset \mathbb{R}^n, a_i, b_i \in \mathbb{R}$, $a_i \leq b_i$, and F is continuous. Further let*

$$\forall j \in \mathbb{N} \text{ and } \forall B \in \mathcal{P}(Q, j): \qquad P\left(\exists l_B \in \mathbb{N} \ : \ P_{l_B} \cap B \neq \emptyset\right) = 1 \tag{6.6.26}$$

Then an application of the algorithm DRA II leads to a sequence of archives $\{A_i\}_{i\in\mathbb{N}}$, such that

$$\lim_{i\to\infty} \text{dist}(F(P_Q), F(A_i)) = 0 \quad \textit{with probability one,}$$

where P_Q denotes the Pareto set of the given MOP.

Proof: Let $x \in P_Q$. Since (6.6.26) holds, for every $i \in \mathbb{N}$ there exists a point $x_i \in B(x, i)$ such that there is with probability one a $j_i \in \mathbb{N}$ with $x_i \in P_{j_i}$ (i.e. the box $B(x, i)$ gets "'visited"' by *generate()* with probability one after finitely many steps). Hence there exists a point $d_i \in A_{j_i}$ with $F(d_i) \leq_p F(x_i)$. By construction of the archives, for all $N > j_i$ there is a point $d_i^N \in A_N$ with

$$F(d_i^N) \leq_p F(d_i). \tag{6.6.27}$$

Since $\lim_{i\to\infty} x_i = x$ and F is continuous it follows that $\lim_{i\to\infty} F(x_i) = F(x)$. Further, since $x \in P_Q$ we can deduce that

$$\lim_{i\to\infty} F(d_i) = F(x). \tag{6.6.28}$$

Combining (6.6.27) and (6.6.28) it follows that

$$\lim_{i\to\infty} \text{dist}(F(x), F(A_i)) = 0 \quad \text{with probability one,}$$

and we are done.

REMARK 6.6.7 Crucial for the convergence of the algorithm DRA II is the condition (6.6.26). In case a MOEA is used for the process *generate()*, this property should easily be provided if a suitable mutation strategy is applied and if the family of (finite) state spaces S_j, $j \in \mathbb{N}$, for the populations P_j are e.g. characterized by

- $S_j \subset Q$

- $S_j \supset S_{j-1}$, if $j \geq 1$, and
- $\forall B \in \mathcal{P}(Q, j) : \exists\, x \in S_j \cap B$

The next example shows that weak Pareto points which are not properly Pareto optimal can cause problems for the convergence of the A_j's toward the Pareto set if the state space of the MOP is continuous.

EXAMPLE 6.6.8 Consider the bicriteria optimization problem which is illustrated in Figure 6.25. Once the weak Pareto point x_1 is added to the archive, this point will only be discarded when x_2 is taken into account, since x_2 is the only point which dominates x_1.

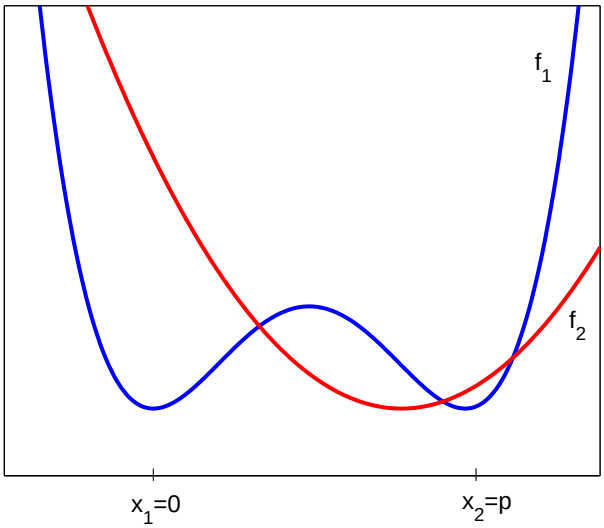


Figure 6.25: The weak Pareto point x_1 is only dominated by x_2.

THEOREM 6.6.9 *In addition to the assumptions of Theorem 6.6.6 assume that there is no weak Pareto point in $Q \backslash P_Q$.*
Then the algorithm described above generates a sequence of archives $\{A_i\}_{i \in \mathbb{N}}$, such that

$$\lim_{i \to \infty} d(F(P_Q), F(A_i)) = 0 \quad \textit{with probability one,}$$

where $d(\cdot, \cdot)$ denotes the Hausdorff distance.

Proof: Using Theorem 6.6.6 it remains to show that

$$\lim_{i \to \infty} \operatorname{dist}(F(A_i), F(P_Q)) = 0 \quad \text{with probability one.} \tag{6.6.29}$$

To see this let $x \in Q \backslash P_Q$. Since x is no weak Pareto point there exists a point $p \in P_Q$ such that $F(p) <_p F(x)$. Since F is continuous there exists a neighborhood $\mathcal{U}(p)$ of p with

$$F(y) <_p F(x) \quad \forall y \in \mathcal{U}(p).$$

Further there exists a $j_p \in \mathbb{N}$ with $B(p, j_p) \subset \mathcal{U}(p)$. Since (6.6.26) holds, there exists with probability one a point $d \in B(p, j_p)$ and an index $j \in \mathbb{N}$ with $d \in P_j$. By this it follows that

$$x \notin A_N \quad \forall N \geq j \quad \text{with probability one,}$$

and the proof is complete.

REMARK 6.6.10 It is known that one problem of the application of MOEAs with elitism is to achieve a satisfying distribution of the entries of the archive (see [102] or [75]). It was observed that this problem is reduced by using the boxes as a tool for the localization of the desired area and for further search; see Figure 6.27 for a (trivial) example. The above condition $(*)$ ensures that the solutions in the archive A_i do not seperate into clusters. In addition, the number of entries in an archive which are contained in a box B can be restricted.

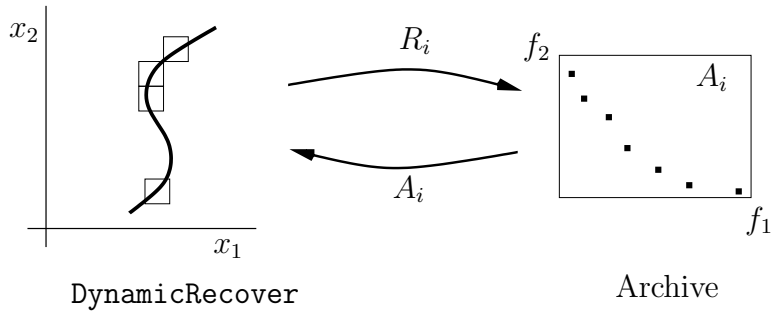


Figure 6.26: Scheme of algorithm `DynamicRecover with Archiving (DRA)`.

6.7 Numerical Results for Non–Smooth Models

Here we present two MOPs which were computed by the algorithms described in the last section. The objectives of both models contain a huge number of local minima which makes them hard to solve by "gradient–based" algorithms.

6.7.1 Example N1

Now we consider the following MOP

$$\begin{aligned} & f_1, f_2 : [-5.12, 5.12]^n \to \mathbb{R} \\ & f_1(x) = \sum_{i=1}^{n-1} (-10 \mathrm{e}^{-0.2\sqrt{x_i^2 + x_{i+1}^2}}) \\ & f_2(x) = \sum_{i=1}^{n} (|x_i|^{0.8} + 5 \sin(x_i)^3) \end{aligned} \tag{6.7.30}$$

Figure 6.28 shows a final population using SPEA (size of initial population: 200; number of generations: 300) and the local improvement by an application of DynamicRecover on this result.

6.7.2 Example N2

Next we consider an MOP which arises in antenna design ([67]):

$$\min \begin{pmatrix} -4\pi^2 \left| \sum_{\nu=-n}^{n} (-i)^\nu \mathcal{J}_\nu(k)(x_\nu + iy_\nu) \right|^2 \\ \max_{\eta=0,..,5} \left(4\pi^2 \left| \sum_{\nu=-n}^{n} (-i)^\nu \mathcal{J}_\nu(k)(x_\nu + iy_\nu) e^{i\nu s_\eta} \right| \right) \end{pmatrix} \tag{6.7.31}$$

subject to the constraints

$$\begin{aligned} x_\nu, y_\nu \in \mathbb{R} \quad (\nu \in \mathbb{Z},\ |z| \le n), \\ 2\pi \sum_{\nu=-n}^{n} (x_\nu^2 + y_\nu^2) \le 1 \end{aligned} \tag{6.7.32}$$

with the special discretization points $s_\eta = \frac{3}{4}\pi + \eta\frac{\pi}{10}$. Here $\mathcal{J}_\nu$ denotes the Bessel function of ν-th order (hence the antenna was modeled by a hollow cylinder). We have tested the model for $n = 5$ and $k = 10$. Since $\mathcal{J}_\nu(x) = (-1)^\nu \mathcal{J}_{-\nu}(x)$ and $\mathbb{C} \cong \mathbb{R}^2$ this leads to a model with 12 (real) free parameters. We have applied recovery techniques on two different SPEA results. Figure 6.29 shows an application on SPEA result with 200 initial individuals and 300 generations. Figure 6.30 for 1000 initial individuals and 500 generations (total running time: 7.5 hours). A comparison of both results shows that the recovering techniques improve existing results but these improvements are local.

Conclusion and Future Work

We have presented algorithms for the computation of the Pareto set of a given multi–objective optimization problem. Furthermore, we have discussed how to combine them to increase the performance and have shown its efficiency on several examples. In future work we have to improve the design of the MOEAs for the special requirements of the different algorithms. Further other optimization metaheuristics like partical swarm optimization have to be tested for the applicability of our methods.

6.8 Extensions for Smooth Models

6.8.1 Introduction

In this section we present improvements of the recovering techniques (see Section 6.3) which can be made when the underlying MOP is smooth enough.
Recall that the substationary points of an MOP are contained in the zero set of

the function $\tilde{F}$ which is defined in (6.2.4). If the objectives – in case the MOP is constrained, also the constraints – are twice continuously differentiable, the tangent space $T_z\mathcal{M}$ (where $\mathcal{M} = \tilde{F}^{-1}(0)$ and $\tilde{F}(z) = 0$) can be computed in a numerically stable way. Using the tangent space of $\mathcal{M}$ at a substationary point $z = (x, \alpha, \lambda)$ with $x \in B_x$, *predictors* $\bar{z} = (\bar{x}, \bar{\alpha}, \bar{\lambda})$ of further substationary points can be generated where $\bar{x}$ is contained in a neighbor box of B_x. Since we use a Gauss–Newton method starting with these predictors to obtain new zeros of $\tilde{F}$, the following recovering techniques can be viewed as a variant of a *predictor–corrector method* for the computation of general implicitly defined manifolds. The boxes serve as a tool for the (uniform) spread of solutions of $\mathcal{M}$. We will see in the next section that the particular advantage of the data structure is that also higher dimensional problems can be solved.

There exist some methods for the computation of differentiable m-manifolds. For $m = 1$ these are the well known *pathfollowing methods* (see e.g. [2] or [98] and references therein). For $m > 1$ there exist the *moving frame algorithms* (see [99], [97], [8] or [61]), the *piecewise linear algorithms* (see [3], [4] and [46]) and the method presented in [56]. In the context of multi–objective optimization homotopy techniques have been considered in [96] and [58].

6.8.2 The Algorithms

Here we propose two algorithms for the computation of smooth multi–objective optimization problems. The aim of the first algorithm is the approximation of the connected components of substationary points which contain "representatives" in an initial box collection $\mathcal{B}$. It can also be applied to the computation of general implicitly defined manifolds. The second algorithm uses in addition a nondominated sorting strategy in order to compute the set of points which are globally optimal – according to the box collection which is generated by the procedure.

To reduce the computational effort we associate with every box B which is added to the collection $\mathcal{B}$ an approximation $a^B \in \mathbb{R}^{n+m+1}$ of a substationary point[12]

$$a^B = (x^B, \alpha^B, \lambda^B) \quad \text{and} \quad \tilde{F}(x^B, \lambda^B, \alpha^B) \approx 0,$$

where $x^B \in B$, which is being computed during the course of the algorithm.
Given a box collection $\mathcal{B}$ containing the corresponding boxes of some initial substationary points of the underlying MOP and a depth d the algorithm CONT–Recover reads as follows:

Algorithm CONT–Recover

(1) mark all boxes $B \in \mathcal{B}$.

(2) for all marked boxes $B \in \mathcal{B}$:

[12] To be more precise, we only allow the addition of a box to the collection if it contains a substationary point.

(a) unmark box

(b) compute a set of orthonormal vectors $\{q_1, \ldots, q_{k-1}\}$ such that $span\{q_1, \ldots, q_{k-1}\} = T_{(x^B, \alpha^B, \lambda^B)}\mathcal{M}$.

(c) generate predictors $s_1, \ldots, s_{n_B} \in T_{(x^B, \alpha^B, \lambda^B)}\mathcal{M}$.

(d) for $i = 1, \ldots, n_B$:
starting with s_i, compute $(x^F, \alpha^F, \lambda^F)$ with $\tilde{F}(x^F, \alpha^F, \lambda^F) \approx 0$.
If $B(x^F, d) \notin \mathcal{B}$: add $B(x^F, d)$ to the collection $\mathcal{B}$, mark the box, and set $a^{B(x^F,d)} := (x^F, \alpha^F, \lambda^F)$.

Repeat **(2)** while new boxes are added to $\mathcal{B}$ or until a prescribed maximal number of steps is reached.

Before we can state the next algorithm we have to make several

REMARKS 6.8.1 (a) The orthonormal vectors $q_1, \ldots, q_{k-1}$ can be computed via a QR-decomposition of $\tilde{F}'$, for details see [58]. If $\dim T_{(x^B, \alpha^B, \lambda^B)}\mathcal{M} < k - 1$, then $\mathcal{M}$ is not a $(k-1)$-dimensional manifold in the neighborhood of the substationary point $(x^B, \alpha^B, \lambda^B)$. In this case we continue the search in all coordinate directions, i.e. we take $q_i = e_i$, $i = 1, \ldots, n$, where e_i is the ith unit vector of $\mathbb{R}^n$.

(b) A note on the predictor step: for an approximation $a^B = (x^B, \alpha^B, \lambda^B)$, which is associated to a box B, we select predictors $s = (x^s, \alpha^s, \lambda^s)$ where x^s is contained in a neighbor box of B, i.e. in a box $\hat{B}$ with $B \cap \hat{B} \neq \emptyset$.

(c) A note on the corrector step: for unconstrained MOPs we have used the iteration function P which is defined in (6.2.8) due to its global convergence. In the constrained case we have used a Gauss–Newton method (see e.g. [33]).

(d) Since no particular structure of $\tilde{F}$ is taken into account the algorithm is able to compute general implicitly defined sets $H^{-1}(0)$ of functions

$$H : \mathbb{R}^n \to \mathbb{R}^m.$$

In case H is only continuous the algorithm stated above can still be applied successfully using the following modifications:

- the predictors can be chosen e.g. as $q_i = e_i$ as described in part (a) of this remark.
- for the corrector any derivative free minimization algorithm applied on $\|H(x)\|$ can serve as e.g. the *downhill simplex method* of Nelder and Mead, see [87] or [7].

By the following obvious estimation it follows that the current box collection $\mathcal{B}$ and all neighboring boxes of $\mathcal{B}$[13] are contained in the basin of attraction of the corrector[14] – if all boxes are sufficiently small.

[13] That is, all boxes $\hat{B} \in \mathcal{P}_k$ such that there exists a $B \in \mathcal{B}$ with $\hat{B} \cap B \neq \emptyset$.

[14] Here we assume that the corrector converges locally. This is e.g. the case when a Gauss-Newton method ([33]) is used.

FACT 6.8.2 Let $B = (c, r) \in \mathcal{B}_k$ and let there exist a substationary point $p \in B$, i.e. there exists an $\bar{\alpha} \in \mathbb{R}^k$ with $\bar{\alpha}_i \geq 0$ and $\sum_{i=1}^k \bar{\alpha}_i = 1$ and $\bar{\lambda} \in \mathbb{R}^m$ such that $\tilde{F}(p, \bar{\alpha}, \bar{\lambda}) = 0$. Furthermore, let $\tilde{F}$ be Lipschitz continuous on a neighborhood $\mathcal{U}$ of $B \times \{\bar{\alpha}\} \times \{\bar{\lambda}\} \subset \mathbb{R}^{n+k+m}$:

$$\|\tilde{F}(x^1, \alpha^1, \lambda^1) - \tilde{F}(x^2, \alpha^2, \lambda^2)\|_2 \leq \mathcal{L}\|(x^1, \alpha^1, \lambda^1) - (x^2, \alpha^2, \lambda^2)\|_2 \quad \forall (x^1, \alpha^1, \lambda^1), (x^2, \alpha^2, \lambda^2) \in \mathcal{U}.$$

Then

$$\|\tilde{F}(x, \bar{\alpha}, \bar{\lambda})\|_2 \leq \mathcal{L}4\|r\|_2 \quad \forall x \in \{x \in \mathbb{R}^n | B \cap B(x, k) \neq \emptyset\}$$

Proof: Let $x \in \{x \in \mathbb{R}^n | B \cap B(x, k) \neq \emptyset\}$. Then

$$\begin{aligned}\|\tilde{F}(x, \bar{\alpha}, \bar{\lambda})\|_2 &= \|\tilde{F}(x, \bar{\alpha}, \bar{\lambda}) - \tilde{F}(p, \bar{\alpha}, \bar{\lambda})\|_2 \\ &\leq \mathcal{L}\|(x, \bar{\alpha}, \bar{\lambda}) - (p, \bar{\alpha}, \bar{\lambda})\|_2 \leq \mathcal{L}\|x - p\|_2 \leq \mathcal{L}4\|r\|_2.\end{aligned}$$

The algorithm CONT–Recover generates a box collection which covers *all* substationary points which can be reached by the continuation process starting with the initial box collection $\mathcal{B}$. In the context of multi–objective optimization this leads to two problems. First, the task is to compute substationary points on every connected component of $\mathcal{M}$. Therefore, typically any (global) multi–objective optimization algorithm can serve. Unfortunately, it is nearly impossible to compute some optimal solutions on every connected compontent – without computing the entire Pareto set – and thus it is reasonable that $\mathcal{B}$ contains points which are not globally optimal[15]. Furthermore, it can occur that even within one connected component of $\mathcal{M}$ there exist global and local Pareto points (see e.g. Example S4). Hence the second problem is to discard the obtained solutions which are not globally optimal. Since it is not efficient to compute every connected component first and then to evaluate the resulting domain we propose to combine the continuation method with a *nondominated sorting* strategy in the following way:
First we build up an archive $\mathcal{A}$ using the representative points x^B of the initial box collection $\mathcal{B}$. In the further recovering steps every "candidate box" B which is computed in step (2d) of the preceeding algorithm is added to the box collection if it can compete with all boxes in $\mathcal{B}$, i.e. if x^B is not dominated by any vector in $\mathcal{A}$ according to the underlying MOP. This has to be continued until no more boxes are added to $\mathcal{B}$ or a prescribed number of steps is reached. Finally, all boxes have to be rechecked because it it possible that $\mathcal{B}$ contains boxes which cannot compete with boxes which were added later to the collection. Afterwards, the algorithm has hopefully computed a suitable covering of the Pareto set. Subsequently, the archive $\mathcal{A}$ can be viewed as a (point–wise) representation of the box collection and thus as a discretization of the Pareto set.

Given a box collection $\mathcal{B}$ and a depth d the algorithm ND–CONT reads as follows:

[15]To be more precise: $\mathcal{B}$ can contain boxes which itself contain no Pareto optimal solutions.

ND–Cont

(1) (a) $\mathcal{A} := \emptyset$

(b) for all boxes $B \in \mathcal{B}$:

mark box

update the archive $\mathcal{A}$ by $(x^B, F(x^B))$

(2) (i) for all marked boxes $B \in \mathcal{B}$:

(a) unmark box

(b) compute a set of orthonormal vectors $\{q_1, \ldots, q_{k-1}\}$ such that $span\{q_1, \ldots, q_{k-1}\} = T_{(x^B, \alpha^B, \lambda^B)}\mathcal{M}$.

(c) generate predictors $s_1, \ldots, s_{n_B} \in T_{(x^B, \alpha^B, \lambda^B)}\mathcal{M}$.

(d) for $i = 1, \ldots, n_B$:

starting with s_i, compute $(x^F, \alpha^F, \lambda^F)$ with $\tilde{F}(x^F, \alpha^F, \lambda^F) \approx 0$. If $B(x^F, d) \notin \mathcal{B}$: add $B(x^F, d)$ to the collection $\mathcal{B}$, mark the box, set $a^{B(x^F,d)} := (x^F, \alpha^F, \lambda^F)$, and update the archive $\mathcal{A}$ by $(x^F, F(x^F))$.

(ii) for all marked boxes $B \in \mathcal{B}$:

if $(x^B, F(x^B)) \notin \mathcal{A}$: $\mathcal{B} := \mathcal{B} \backslash B$

Repeat **(2)** while new boxes are added to $\mathcal{B}$ or until a prescribed maximal number of steps is reached.

(3) for all boxes $B \in \mathcal{B}$:

if $(x^B, F(x^B)) \notin \mathcal{A}$: $\mathcal{B} := \mathcal{B} \backslash B$

When the MOP is unconstrained we take in addition to x^B further sample points of a box B to stabilize the selection process. In this case ND–Cont can be viewed as a combination of the Recover algorithm and the Sampling algorithm which are described in Section 6.3.

6.8.3 Uniform Distribution of Solutions

The fundamental task of all algorithms presented in this chapter is to provide the decision maker with a sufficient survey of the set of Pareto points. Since the archive remains finite, the request is that the solutions which are stored in the archive are distributed uniformly and do not resolve into several clusters. In the following we give a strategy for the adaptive choice of the size of the boxes which are added to the collection $\mathcal{B}$, since $\mathcal{B}$ is the basis for the archive.

The adaption of the box size is motivated by the following estimation.

FACT 6.8.3 Let $x^* \in B_{x^*} = (c, r) \in \mathcal{B}$ be a substationary point and let the radii of all neigboring boxes of B_{x^*} be equal to r. Furthermore, if F is Lipschitz continuous with Lipschitz constant L, the following estimation holds:

$$\|F(y) - F(x^*)\|_2 \leq 4\|r\|_2 L \qquad \forall y \in B \in \mathcal{B} : \; B \cap B_{x^*} \neq \emptyset \tag{6.8.33}$$

If a prescibed tolerance can be given for every objective – e.g. the smallest change of the function value which is physically relevant – the size of box $B_{x^*} = (c, r)$ which contains the substationary point x^* of the *rescaled*[16] problem can be chosen as follows:

$$\|r\|_2 \approx \frac{tol}{4L} \tag{6.8.34}$$

If the boxes are small enough, one may estimate the Lipschitz constant L by $\|DF(c)\|$, which is similar in spirit to [58].

EXAMPLE 6.8.4 Let us consider the following MOP :

$$\begin{gathered} \min_{x \in \mathbb{R}^2} F(x) \\ F(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \end{pmatrix} = \begin{pmatrix} (x_1 - 1)^4 + (x_2 - 1)^4 \\ (x_1 + 1)^2 + (x_2 + 1)^2 \end{pmatrix} \end{gathered} \tag{6.8.35}$$

The Pareto set of MOP (6.8.35) is given by

$$\mathcal{P} = \left\{ \lambda \begin{pmatrix} -1 \\ -1 \end{pmatrix} + (1 - \lambda) \begin{pmatrix} 1 \\ 1 \end{pmatrix} : \lambda \in [0, 1] \right\}.$$

Figure 6.32 shows two different discretizations of $\mathcal{P}$. In Figure 6.32 (a) the Pareto set is approximated by points $x_i, i = 1, \ldots, N$, which are placed equidistant in parameter space:

$$x_i = \begin{pmatrix} -1 \\ -1 \end{pmatrix} + \frac{2i}{N} \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}.$$

Next, the Pareto set was discretized using the adaptive step size control which is proposed above:

$$x_0 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \quad x_{i+1} = x_i + h_i \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix},$$

where $h_i = \frac{tol}{\tilde{L}_i}$ and

$$\tilde{L}_i := \|DF(x_i)\|_\infty = \max\{\|\nabla f_1(x_i)\|_1, \|\nabla f_2(x_i)\|_1\}.$$

Figure 6.32 (b) shows the discretization points x_i for $tol = 1$ yielding a satisfying distribution of the solution set.

Having computed an approximation of the entire Pareto set $\mathcal{P}$ using the strategy described above the so–called *knee* of $\mathcal{P}$ can easily be determined. The knee of a Pareto set will be defined here[17] as follows:

$$K(\mathcal{P}) := \min_{x \in \mathcal{P}} \|DF(x)\|_\infty = \min_{x \in \mathcal{P}} (\max\{\|\nabla f_1(x)\|_1, \|\nabla f_2(x)\|_1\}). \tag{6.8.36}$$

[16] The important rescalization problem and the related problem of finding appropriate norms for the treatment of MOPs will not be discussed in this work. For this we refer to [78].

[17] An alternative definition of the knee can be found in [17].

For bicriteria optimization problems $K(\mathcal{P})$ can be interpreted as the maximal bulge of the solution curve. The knee is interesting in applications because it serves quite often as the starting point of the decision making process. Figure 6.33 shows the values of $\|DF(x)\|_\infty$ along the Pareto set $\mathcal{P}$ of MOP (6.8.35) as well as $K(\mathcal{P})$. It has to be mentioned that $K(\mathcal{P})$ is not invariant under the scalarization of the objectives. However, this seems to be in the nature of the decision making problem for the author's opinion.

6.9 Numerical Results for Smooth Models

In this section we illustrate the efficiency of the algorithms which are constructed for MOPs which are at least twice continuously differentiable by several examples.

6.9.1 Example S1

First we consider the following unconstrained MOP:

$$\begin{aligned} & f_1, f_2, f_3 : \mathbb{R}^n \to \mathbb{R} \\ & f_i(x) = \sum_{\substack{j=1 \\ j \neq i}}^{n} (x_j - a_j^i)^2 + (x_i - a_i^i)^4, \end{aligned} \tag{6.9.37}$$

where

$$\begin{array}{lcll} a^1 & = & (1,1,1,1,\ldots) & \in \mathbb{R}^n \\ a^2 & = & (-1,-1,-1,-1,\ldots) & \in \mathbb{R}^n \\ a^3 & = & (1,-1,1,-1,\ldots) & \in \mathbb{R}^n \end{array}$$

Figure 6.34 shows (once more) how the recovering techniques are working. In that case, the algorithm CONT–Recover was started with only two initial boxes which contain substationary points (Figure 6.34 (a)). The algorithm could also be applied successfully on higher dimensional problems, see Figure 6.35.

6.9.2 Example S2

Next we consider a multi–objective optimization problem consisting of four objectives. In fact, it is an augmented model of MOP (6.9.37):

$$\begin{aligned} & f_1, f_2, f_3, f_4 : \mathbb{R}^n \to \mathbb{R} \\ & f_i(x) = \sum_{\substack{j=1 \\ j \neq i}}^{n} (x_j - a_j^i)^2 + (x_i - a_i^i)^4, \end{aligned} \tag{6.9.38}$$

where

$$\begin{array}{lcll} a^1 & = & (1, 1, 1, 1, \ldots) & \in \mathbb{R}^n \\ a^2 & = & (-1,-1,-1,-1,\ldots) & \in \mathbb{R}^n \\ a^3 & = & (1,-1,1,-1,\ldots) & \in \mathbb{R}^n \\ a^4 & = & (1, 1,-1,-1,\ldots) & \in \mathbb{R}^n \end{array}$$

Figures 6.36 and 6.37 show computational results for dimension $n = 10$ in parameter space and in image space. The latter figure indicates that the decision making process may be a challenging task for a "real world problem". This problem is typically yet hard enough for $k \leq 3$, but for four objectives there seems to miss the general view on the set of optimal solutions, even when they are all computed.

6.9.3 Example S3

The recovering techniques which are presented in the last section can also be applied on general implicitly defined manifolds. In this example we make an insertion from the field of multi–objective optimization and attend to the computation of $H^{-1}(0)$ of a continuously differentiable function $H : \mathbb{R}^n \to \mathbb{R}^m$.
In the following we consider the two functions

$$\begin{aligned} H_1, H_2 &: \mathbb{R}^3 \to \mathbb{R}^1 \\ H_1(x,y,z) &:= x^4 - 3xy - \cos(4z) + \cos(xy) \\ H_2(x,y,z) &:= r^2 - z^2 - (\sqrt{x^2+y^2} - R)^2 \end{aligned} \tag{6.9.39}$$

The coverings of the 2-manifolds $H_1^{-1}(0)$ and $H_2^{-1}(0)$ are shown in Figure 6.38. The recovering techniques can be viewed as a particular version of well–known predictor–corrector methods (see e.g. [97] or [2]). The improvement of this technique will be one topic for the author for further studies.

6.9.4 Example S4

Now we turn our attention to the following constrained MOP:

$$\min F(x) := \begin{pmatrix} (x_1-1)^4 + (x_2-1)^2 + (x_3-1)^2 \\ (x_1+1)^2 + (x_2+1)^4 + (x_3+1)^2 \\ (x_1-1)^2 + (x_2+1)^2 + (x_3-1)^4 \end{pmatrix}$$

subject to the equality constraint

$$h(x) = r^2 - z^2 - (\sqrt{x^2+y^2} - R)^2 = 0,$$

The set of substationary points are thus contained in $\tilde{F}^{-1}(0)$, where

$$\begin{aligned} \tilde{F} &: \mathbb{R}^{3+1+3} \to \mathbb{R}^{3+1+1} \\ \tilde{F}(x,\alpha,\lambda) &:= \begin{pmatrix} \sum_{i=1}^k \alpha_i \nabla f_i(x) + \lambda \nabla h(x) \\ h(x) \\ \sum_{i=1}^k \alpha_i - 1 \end{pmatrix} \end{aligned}$$

Figure 6.39 shows both the set of substationary points (via CONT-Recover) and the Pareto set (via ND-Cont) for the following values:

$$Q = [-1,1]^3, \quad z = 0, \quad r = 0.3, \quad R = 0.5.$$

6.9.5 Example S5

Finally we consider *parameter dependent models* of the following form:

$$\min F_\lambda : \mathbb{R}^n \to \mathbb{R}^k, \qquad \lambda \in \mathbb{R}^d$$

This particular kind of problem e.g. occurs when λ is given data for the underlying system which is modelled by F and can change during the optimization process. (see e.g. [94]) In case λ changes quickly it is not advisable to compute the entire Pareto set for every value of λ but it may be more efficient to approximate the set $\tilde{F}^{-1}(0)$, where

$$\begin{aligned} \tilde{F} &: \mathbb{R}^{n+d+k} \to \mathbb{R}^{n+1} \\ \tilde{F}(x,\lambda,\alpha) &:= \begin{pmatrix} \sum_{i=1}^{k} \alpha_i \frac{\partial f_i}{\partial x}(x,\lambda) \\ \sum_{i=1}^{k} \alpha_i - 1 \end{pmatrix}. \end{aligned} \tag{6.9.40}$$

When the auxiliary system is computed, the set of substationary points for every value $\bar{\lambda}$ are given by the projection $\tilde{F}^{-1}(0)|_{\lambda=\bar{\lambda}}$, which can easily be identified in the corresponding box collection.
Now we consider the following parameter dependent MOP:

$$F_\lambda(x) := (1-\lambda)F_1(x) + \lambda F_2(x), \tag{6.9.41}$$

where

$$\begin{aligned} F_1, F_2 &: \mathbb{R}^2 \to \mathbb{R}^2 \\ F_1(x_1,x_2) &= \begin{pmatrix} (x_1-1)^4 + (x_2-1)^2 \\ (x_1+1)^2 + (x_2+1)^2 \end{pmatrix}, \\ F_2(x_1,x_2) &= \begin{pmatrix} (x_1-1)^2 + (x_2-1)^2 \\ (x_1+1)^2 + (x_2+1)^2 \end{pmatrix}. \end{aligned} \tag{6.9.42}$$

Figure 6.41 shows the set $\tilde{F}^{-1}(0)$ of (6.9.41). Two "classical" Pareto sets particular values of λ – using the according parts of the box collection for the auxiliary system – can be seen in Figure 6.42.

6.10 Conclusion and Future Work

In this chapter we have proposed set oriented methods which aim for the computation of the entire Pareto set of given MOPs with different smoothness assumptions. The algorithms can in principle be divided into two categories. First, there are algorithms which are based on the subdivision techniques presented in Chapter 2. These algorithms are global in nature but restricted to "moderate" dimensions of

the parameter space. Furthermore, there are the recovering algorithms which allow the computation of higher dimensional MOPs. Here, the boxes serve as a surprisingly efficient data structure for the storage and the distribution of the substationary points due to the simplicity of its construction. The drawback of these techniques is that they can perform only locally. In order to increase the total performance – i.e. to obtain robust algorithms for the computaton of the Pareto set which are not restricted to low dimensional MOPs – we have given possible ways to overcome this problem by combining the algorithms and/or by involving archiving stategies.

A comparison of the methods for the numerical treatment of MOPs is a difficult task since the different solutions obtained by these methods are not easy to compare. In the case one can measure the importance of all the objectives of an underlying MOP an algorithm of the (large) class of "scalarizing" methods like e.g. the weighting sum approach has certainly to be preferred. However, in a lot of applications the proper a priori choice of these weights is typically the crucial problem and the main reason why there is a desire to obtain the entire Pareto set.
For the comparison of different approximations of the set of Pareto points of a particular MOP there exists a variety of *performance indices* (see [21] and references therein). Since the author shares the opinion of [89] that many performance indices may be misleading in that they fail to truly reflect the quality of the solution sets we will not use one of these techniques but will try to classify the set oriented algorithms to other existing methods in a more general way.
The subdivision techniques seem to be advantageous in particular for continuous, moderate dimensional MOPs due to the global approach. In case the MOP is not continuous and/or the dimension of the parameter space is large, heuristics like evolutionary algorithms or particle swarm optimization seem to be the best existing methods so far for the approximation of the Pareto set. For these models the recovering techniques can merely be applied to improve the results which are obtained by these heuristic algorithms locally and to improve the diversity of the solution set. This changes when the underlying MOP is smooth enough. In this case the recovering techniques can be used to compute the connected components of the substationary points according to an initial set of solutions very efficiently due to the knowledge of the structure of the solution set. The results which can be obtained by the recovering techniques and the method presented in [18] – also the method described in [43] in case the MOP is convex – seem to be similar for two and for three objectives. For more than three objectives the recovering techniques seem to be the only methods which can produce solutions quickly and reliably.

There are some topics which have to be addressed in further studies. First, there is the general improvement of the algorithms. For instance, some work has to be done for the computation of non–smooth models. Here, a combination of the set oriented methods with other bio-inspired methods like the particle swarm approach seems to be very promising and has to be tested in future. Furthermore, the computation of general implicitly defined manifolds is under investigation and has to be advanced.

Another important challenge is the decision making process which is the next logical step after the computation of the Pareto set. There are examples where the knowledge *and* storage of the entire solution set is desired – see e.g [94] for the solution of a particular online-optimization problem for the construction of mechatronical systems. However, in most applications the decision maker has to decide for *one* "optimal" adjustment of the underlying system after (or during) the optimization process. Recently there have been proposed several software packages for the support of this problem ([76], [118], [25]). Now the question arises how these tools change the view of the decision maker and thus if this changes the demand on the algorithms for the numerical treatment of MOPs. This interesting point has certainly to be addressed.

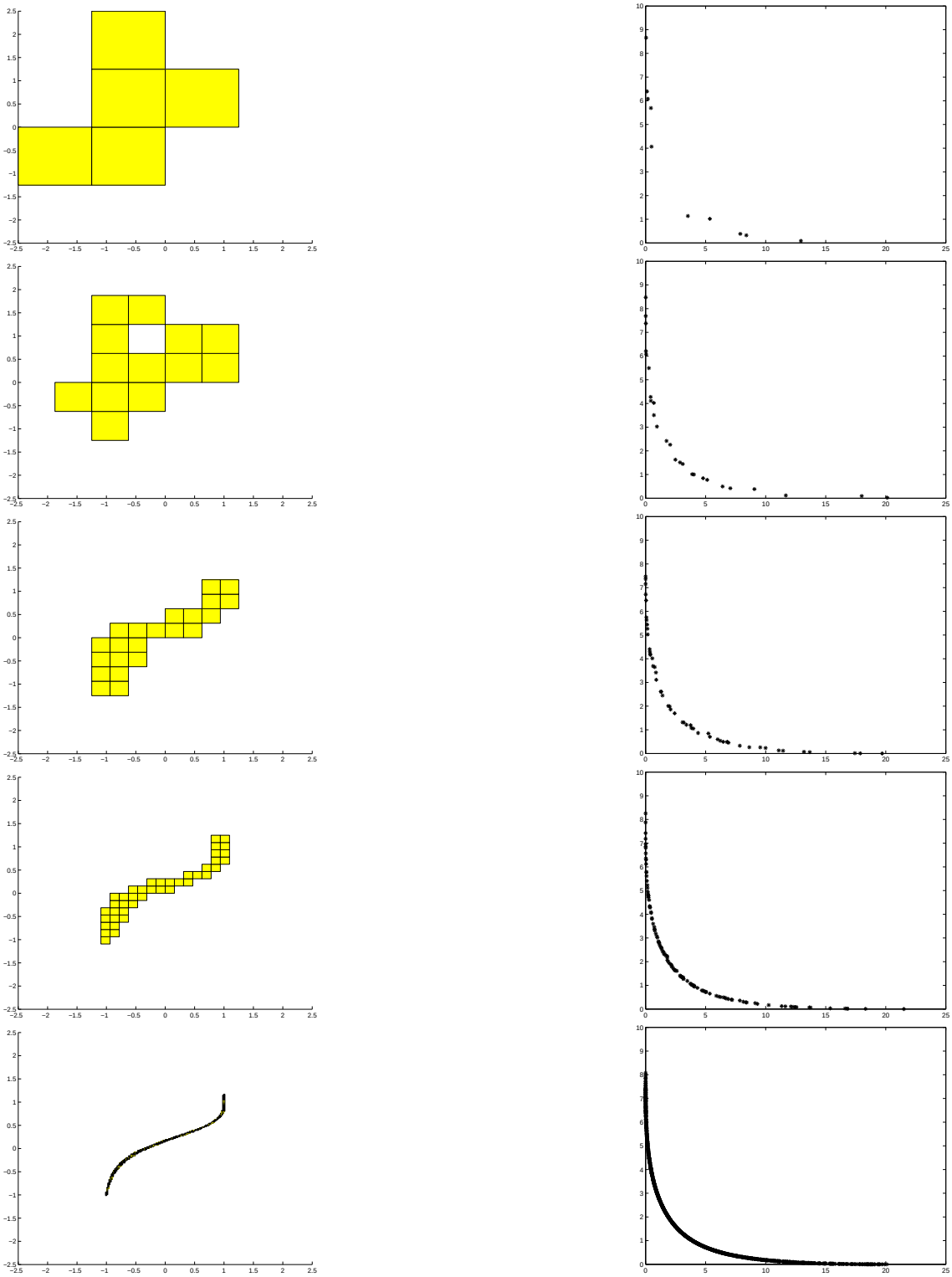


Figure 6.27: Application of algorithm DRA II: the figures show box coverings $\mathcal{B}_i$ and archives A_i of MOP (6.4.12) for $i = 6, 8, 10, 12, 18$.

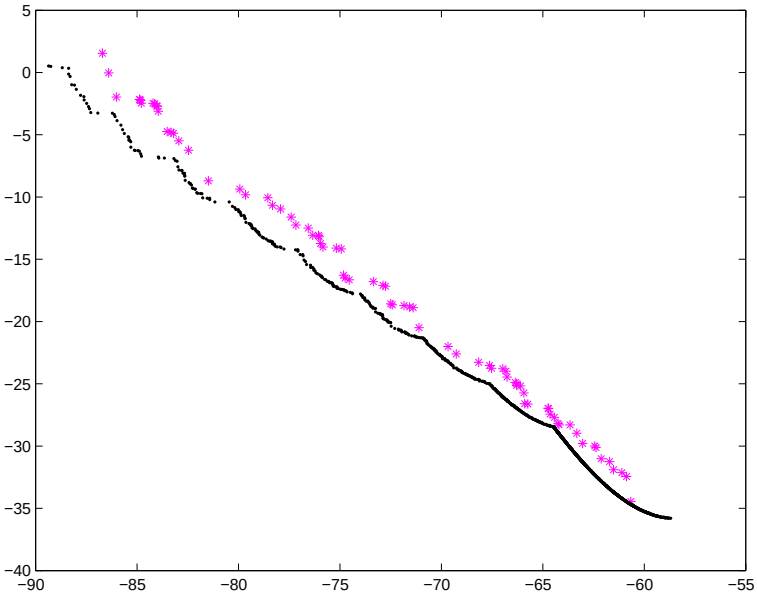


Figure 6.28: Local improvement of MOEA result on MOP (6.7.30) by using DynamicRecover.

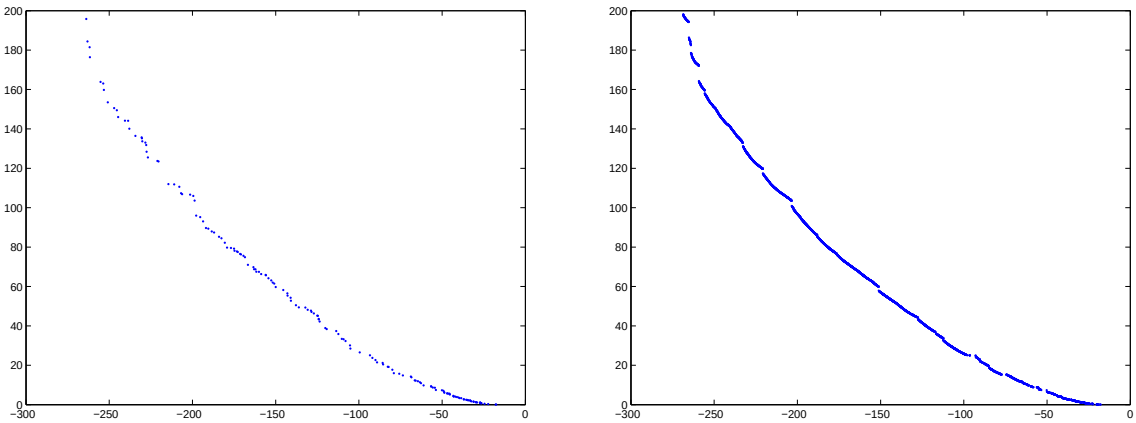
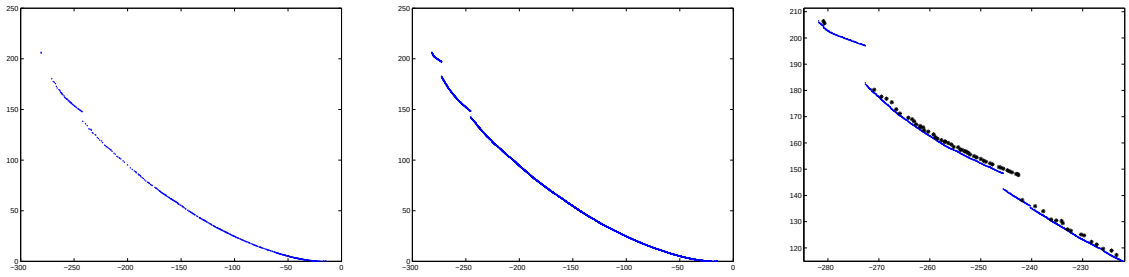
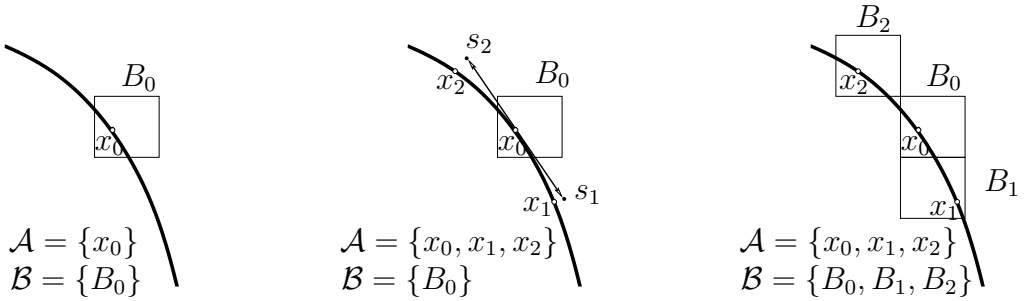


Figure 6.29: Computation of MOP (6.7.31): SPEA result (left) and application of DynamicRecover (right).



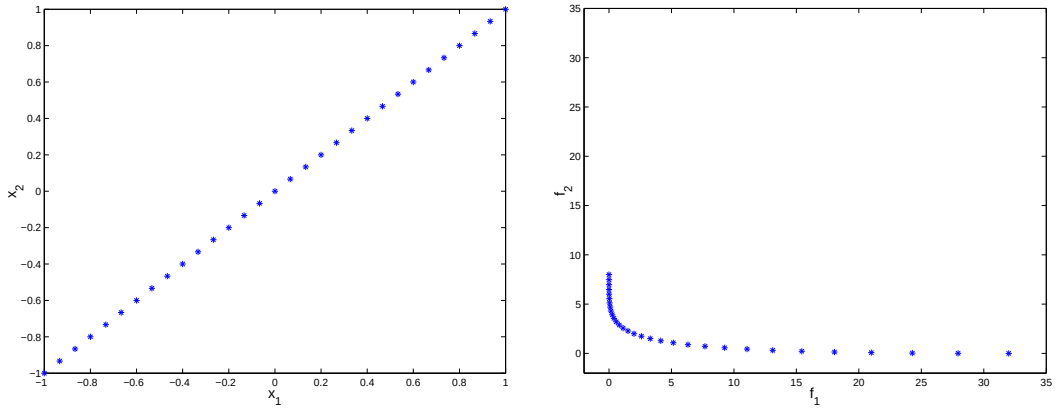
(a) SPEA Result (b) StaticRecover (c) Comparison

Figure 6.30: Computation of MOP (6.7.31): Application of StaticRecover on a SPEA result and a comparison in a selected area.

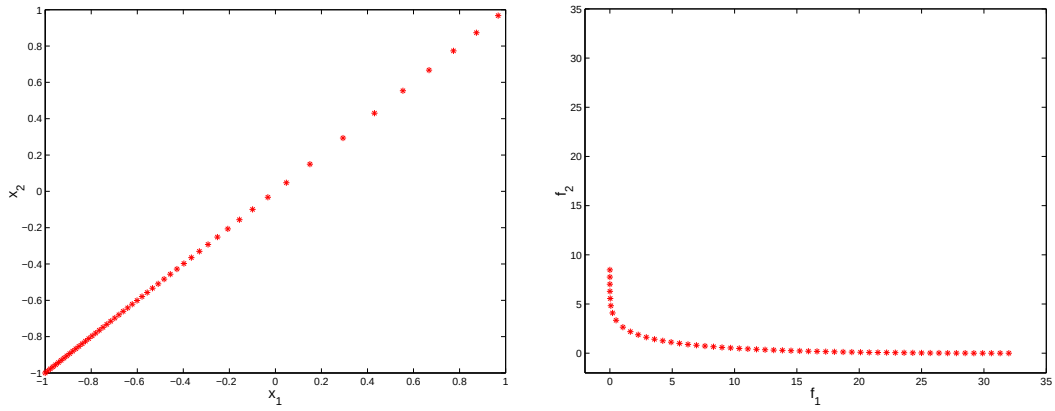


(a) (b) (c)

Figure 6.31: Possible assignment of a box collection $\mathcal{B}$ and archive $\mathcal{A}$ by the use of ND–Cont. The black line indicates the set of substationary points.



(a) fixed step size



(b) adaptive step size

Figure 6.32: Discretizations of the Pareto set of MOP (6.8.35) with (a) fixed step size and (b) adaptive step size control.

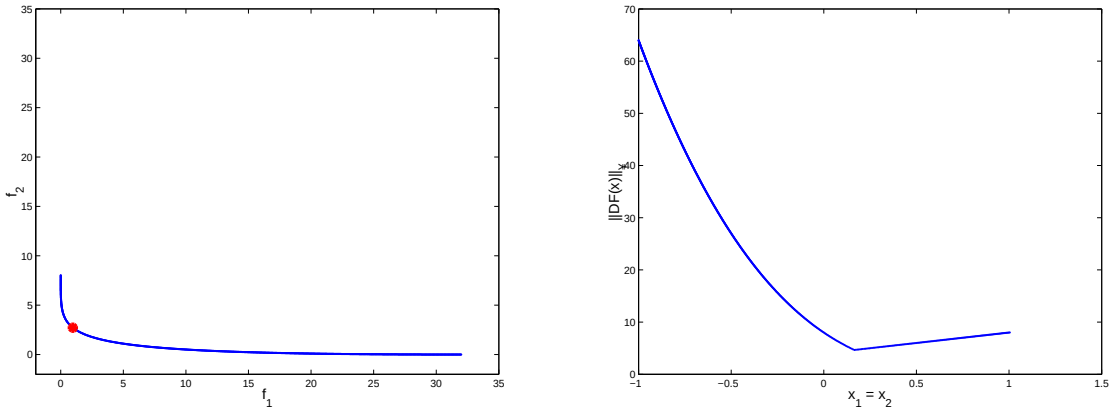
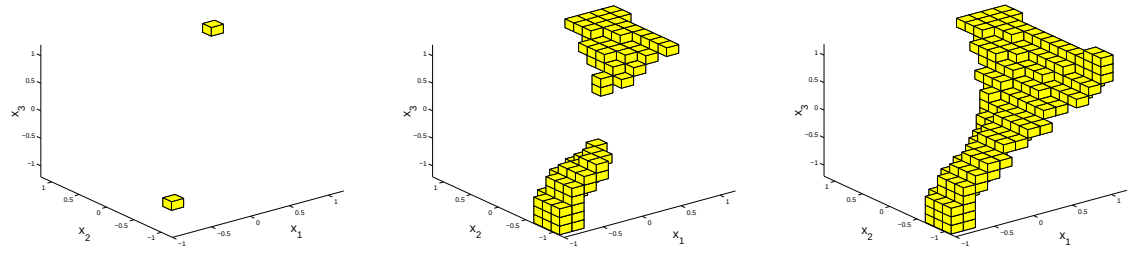
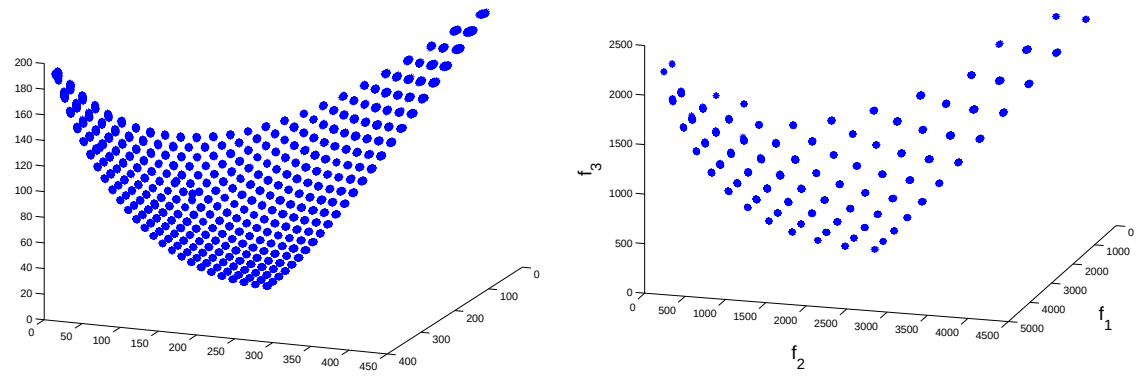


Figure 6.33: The knee of MOP (6.8.35) at $x \approx (0.165, 0.165)^T$.



(a) initial box collection (b) 6 homotopy steps (c) 12 homotopy steps

Figure 6.34: Computation of MOP (6.9.37) for dimension $n = 3$. The figures show the initial box collection and two extensions. The algorithm stops after 12 homotopy steps with a perfect covering of the Pareto set.



(a) Dimension $n = 100$ (b) Dimension $n = 1000$

Figure 6.35: Pareto sets of MOP (6.9.37) in image space.

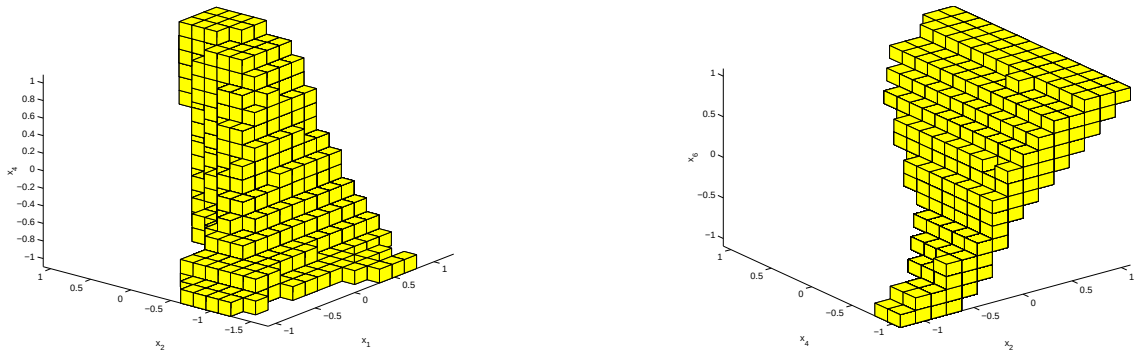


Figure 6.36: Computation of MOP (6.9.38): The figures show two projections of the covering of the Pareto set in parameter space for $n = 10$.

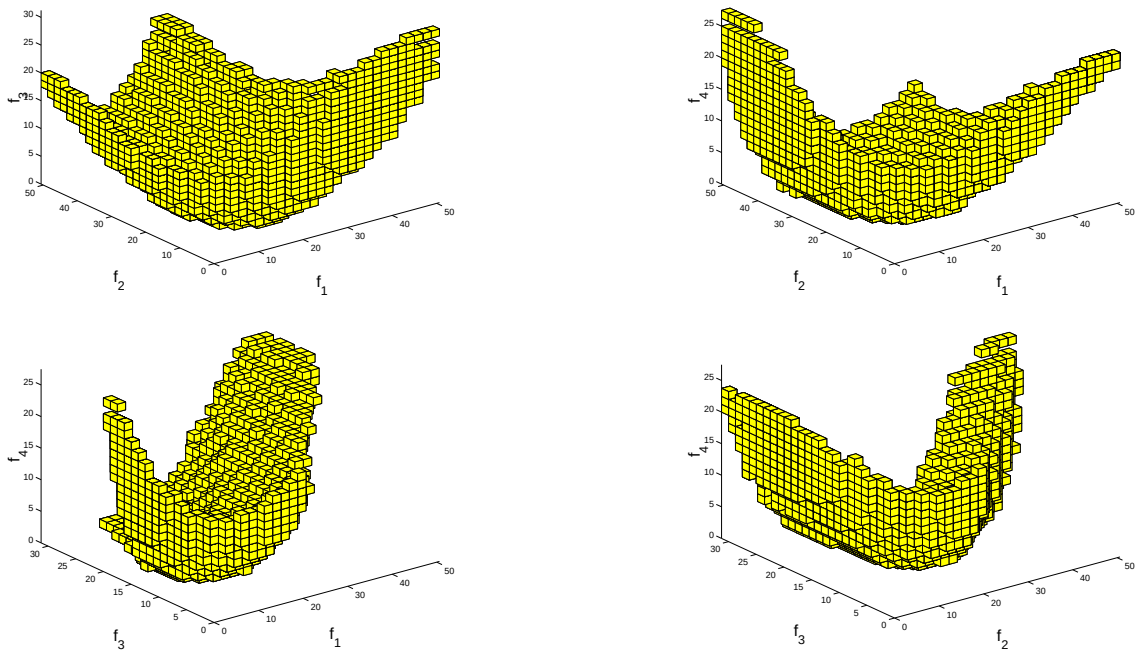
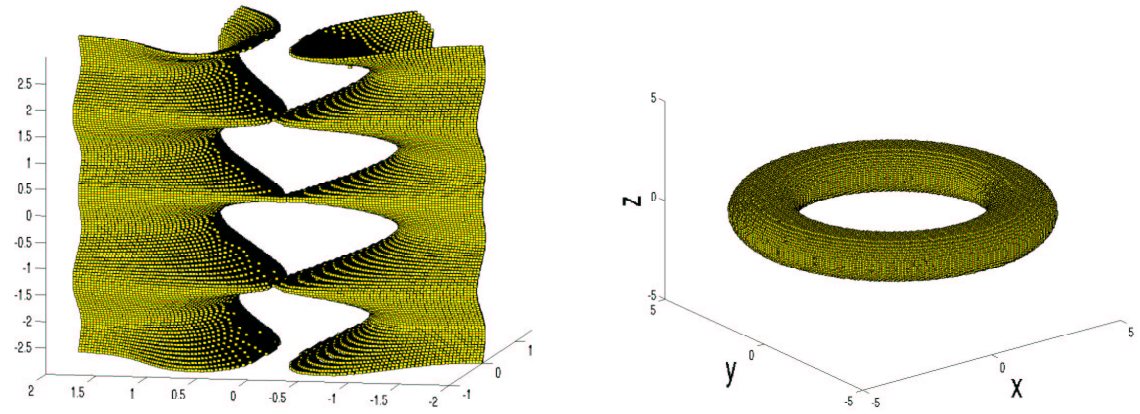


Figure 6.37: Computation of MOP (6.9.38): The figures show all projections of the Pareto set in the image space for $n = 10$.



(a) $H_1^{-1}(0)$ (b) $H_2^{-1}(0)$

Figure 6.38: Computation of implicitly defined manifolds (see (6.9.39)). For H_2 we have chosen $r = 1$ and $R = 4$.

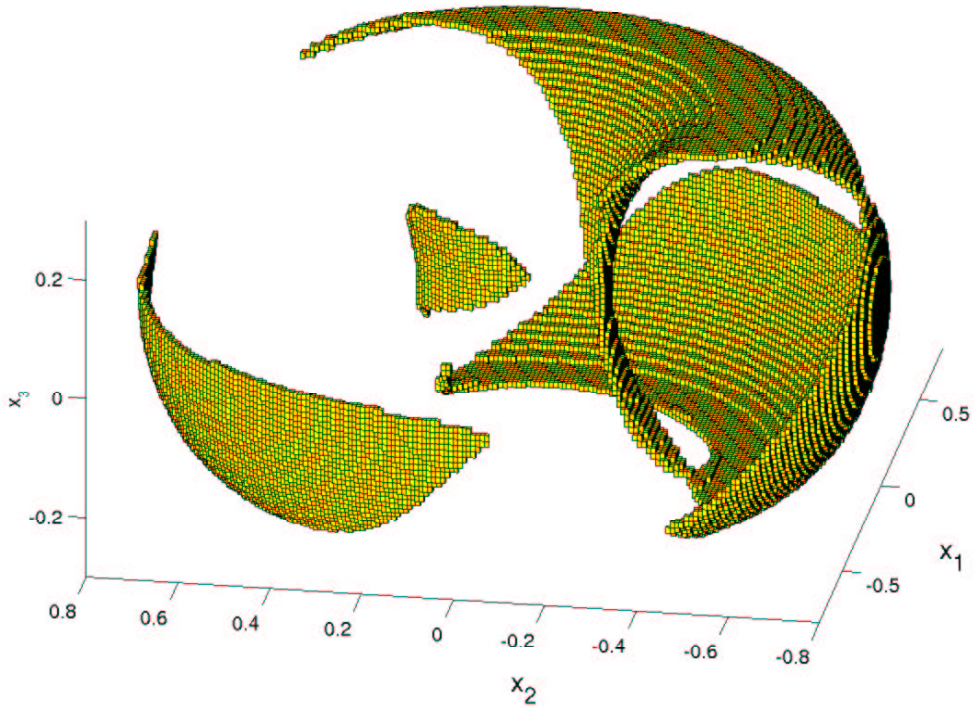
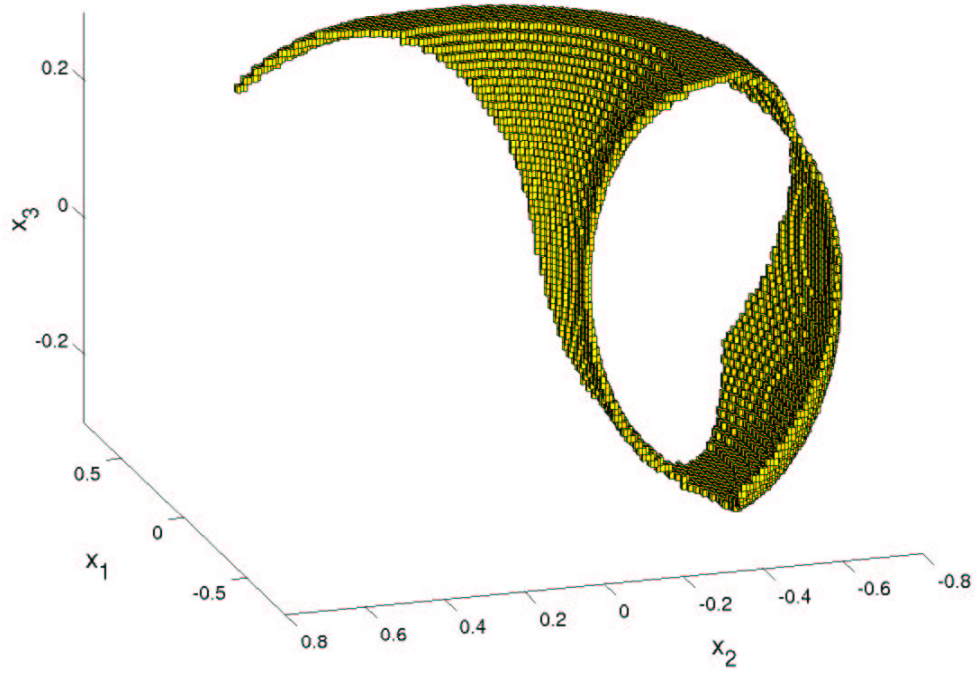


Figure 6.39: Computation of MOP (6.9.4): set of substationary points (above) and Pareto points (below).



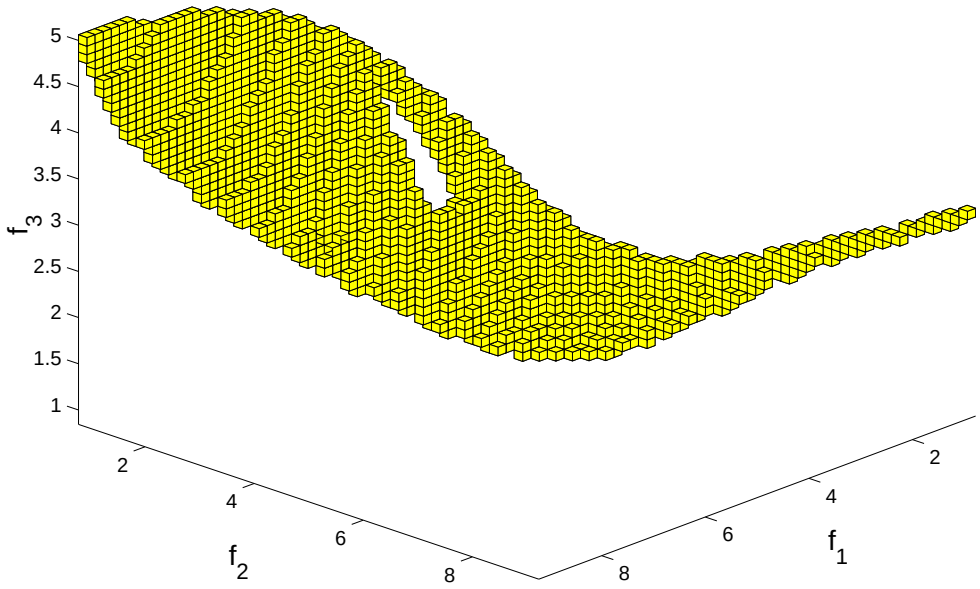


Figure 6.40: Computation of MOP (6.9.4): set of Pareto points in image space.

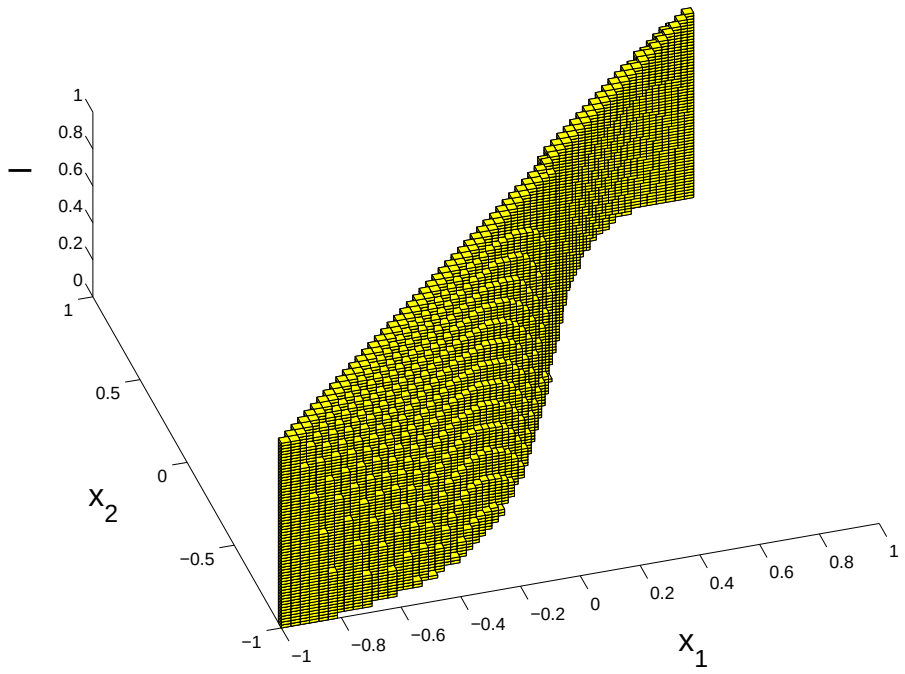
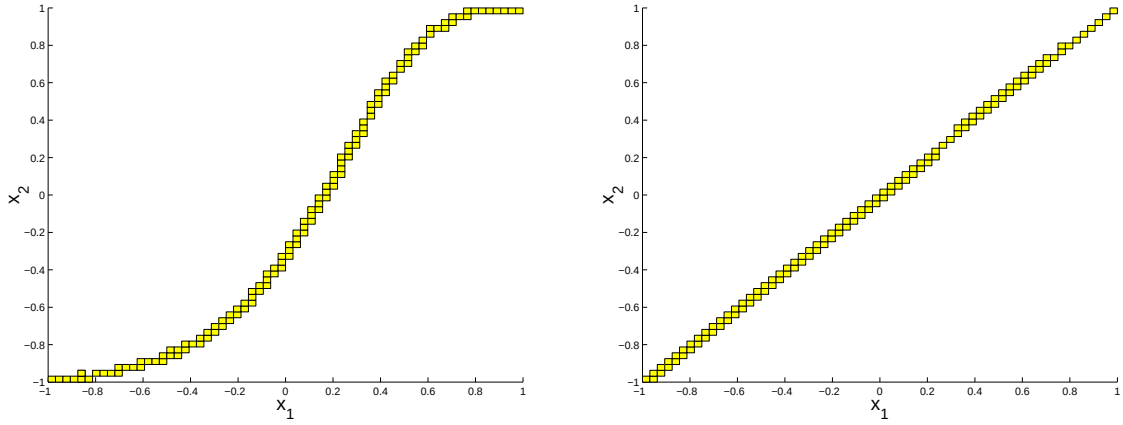


Figure 6.41: Family of Pareto sets, see (6.9.41).



(a) $\lambda = 0$ (b) $\lambda = 1$

Figure 6.42: Pareto sets for two values of λ of MOP (6.9.41).

Chapter 7

Conclusion

In this thesis we have presented a variety of set oriented algorithms for the computation of several well-known classes of (global) optimization problems.
To be more precise, we have developed methods for the localization of roots within a prescribed compact region in $\mathbb{R}^n$ and in particular in $\mathbb{C}$. Based on these results we have derived algorithms for the computation of scalar optimization problems and for the localization of the stability regions of delay differential equations. In the main part of this thesis, we have developed techniques for the computation of multi-objective optimization problems for different assumptions of smoothness on the underlying models – including equality constraints.
Though we have presented a convergence result of the subdivision algorithm in a general and abstract form, the quality of all the different algorithms have mainly been proven on numerous numerical results on models which are of academic nature and also on models which arise in real world applications.
Overall, it turned out that the set oriented methods presented in this thesis are well suited for the computation of many classes of global optimization problems and are not always restricted to a moderate number of dimensions.

The algorithms designed in this thesis were implemented in the software package GAIO[1]. It provides flexible and universal interfaces, in order to be easily used for the application to "real world" problems.

[1] `http:www-math.upb.de/~agdellnitz/gaio`

List of Figures

List of Tables

Bibliography

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations*. Academic Press, 1983.

[2] E. L. Allgower and K. Georg. *Numerical Continuation Methods*. Springer, 1990.

[3] E. L. Allgower and S. Gnutzmann. An algorithm for piecewise linear approximation of implicitly defined two-dimensional surfaces. *SIAM Journal of Numerical Analysis*, 24:452–469, 1987.

[4] E. L. Allgower and P. H. Schmidt. An algorithm for piecewise-linear approximation of an implicitly defined manifold. *SIAM Journal of Numerical Analysis*, 22:322–346, 1987.

[5] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, pages 509–517, 1975.

[6] J. L. Bentley and J. H. Friedmann. Data structures for range searching. *Computing Surveys*, 4:398–409, 1979.

[7] R. P. Brent. *Algorithms for Minimization without Derivatives*. Englewood Cliffs, NJ, 1973.

[8] M. L. Brodzik and W.C. Rheinboldt. The computation of simplicial approximations of implicitly defined two- dimensional manifolds. *Comput. Math. Appl.*, 9:9–21, 1994.

[9] J. Buescu, editor. *Trends in Mathematics.*, chapter I. Stewart and T. Elmhirst and J. Cohen: Symmetry-Breaking as an Origin of Species., pages 3–54. Birkhäuser, 2003.

[10] T. A. Burton. Lyapunov's direct method for delay equations. In *Proceedings of the 11th International Conference on Nonlinear Oszillations*, 1987.

[11] G. Castiglioni, K. Jäker, and F. Schlüter. Das aktive Fahrwerk mit elektrischen Aktuatoren. *AT Automatisierungstechnik*, 07 1996.

[12] I. Chauduri, S. Sertl, H. Zoltán, M. Dellnitz, and T. Fraunheim. Global optimization of silicon nanoclusters. *Applied Surface Science*, 226:108–113.

[13] S. N. Chow, J. Mallet-Paret, and J. A. Yorke. Finding zeros of maps. *Math. Comp.*, 32:887–899, 1978.

[14] C. A. Coello Coello and M. S. Lechunga. MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the IEEE World Congress on Computational Intelligence*, IEEE Press, 2002.

[15] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, 2002.

[16] D. Corne, M. Dorigo, and F. Glover. *New Ideas in Optimization*. Mc Graw Hill, 1999.

[17] I. Das. On characterizing the "knee" of the Pareto curve based on n normal-boundary intersection. *Structural Optimization*, 2/3:107–115, 1999.

[18] I. Das and J. Dennis. *Normal-boundary intersection*: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal of Optimization*, 8:631 – 657, 1998.

[19] M. de Berg, M. van Kreveld, M. Overmars, and O. Scharzkopf. *Computational Geometry: algorithms and applications*. Springer Verlag, 1997.

[20] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[21] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[22] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature VI (PPSN-VI)*, pages 849–858, 2000.

[23] K. Deb, S. Pratap, and A. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858. Springer, 2000.

[24] K. Deimling. *Nichtlineare Gleichungen und Abbildungsgrad*. Springer, 1974.

[25] A. Dell'Aere, O. Schütze, K. Witting, H. Zabel, H. Vöcking, A. Gambuzza, M. Dellnitz, and J. Lückel. ParetoViewer – a new interactive visualization tool for decision making on the solution set of multi.objective optimization problems. in progress, 2005.

[26] M. Dellnitz and A. Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numerische Mathematik*, 75:293–317, 1997.

[27] M. Dellnitz and O. Junge. An adaptive subdivision technique for the approximation of attractors and invariant measures. *Comput. Visual. Sci.*, 1:63–68, 1998.

[28] M. Dellnitz, O. Junge, R. Strzodka, and M. Rumpf. The computation of an invariant set inside a cylinder with a knotted flow. *Proc. of Equadiff99*, pages 1053 – 1059, 2000.

[29] M. Dellnitz, O. Schütze, and T. Hestermeyer. Covering Pareto sets by multilevel subdivision techniques. *Journal of Optimization Theory and Application*, 124(1):113–136, 2004.

[30] M. Dellnitz, O. Schütze, and St. Sertl. Finding zeros by multilevel subdivision techniques. *IMA Journal of Numerical Analysis*, 22(2):167–185, 2002.

[31] M. Dellnitz, O. Schütze, and Q. Zheng. Locating all the zeros of an analytic function in one complex variable. *Journal of Computational and Applied Mathematics*, 138:325–333, 2002.

[32] J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[33] P. Deuflhard and A. Hohmann. *Numerische Mathematik*. de Gruyter, 1991.

[34] R. Devaney. *Chaotic Dynamical Systems*. Addison-Wesley, 1989.

[35] O. Diekmann, S. A. van Gils, S. M. Verduyn Lunel, and H.-O. Walther. *Delay Equations*. Springer, 1995.

[36] R. Elsässer. private communication, University of Paderborn, 2003.

[37] L. E. El'sgol'ts and S. B. Norkin. *Introduction to the Theory and Application of Differential Equations with Deviating Arguments*. Academic Press, 1973.

[38] K. Falconer. *Fractal Geometry*. John Wiley & Sons, 1990.

[39] J. Fieldsend, R.M. Everson, and S. Singh. Using unconstrained elite archives for multiobjective optimisation. *IEEE Trans. Evol. Comp.*, 7(3):305–323, 2003.

[40] J. E. Fieldsend and S. Singh. A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence. In *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, 2002.

[41] R. A. Finkel and J. L. Bentley. Quad trees, a datastructure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[42] D. Fischer, M. Börner, and R. Isermann. Control of mechatronic semi-active vehicle suspensions. In *2nd IFAC Conference on Mechatronic Systems, Berkeley, Ca*. IFAC, 2002.

[43] J. Fliege and A. Heseler. Constructing approximations to the efficient set of convex quadratic multiobjective problems. University of Dortmund, Germany, Technical Report, 2003.

[44] J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical Methods of Operations Research*, 51(3):479–494, 2000.

[45] C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele. *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003.* Springer, 2003.

[46] S. Gnutzmann. *Stückweise lineare Approximation implizit definierter Mannigfaltigkeiten*. PhD thesis, University of Hamburg, 1989.

[47] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.

[48] A. Göpfert and R. Nehse. *Vektoroptimierung*. BSB Teubner Verlagsgesellschaft, Leipzig, 1990.

[49] G. Grübel and H. D. Joos. Multi-objective parameter synthesis (MOPS). In J. F. Magni, S. Bennani, and J. Terlouw, editors, *Robust Flight Control, Lecture Notes in Control and Information Sciences 224*, Springer, London, 1997.

[50] J. Guddat, F. Guerra Vasquez, K. Tammer, and K. Wendler. *Multiobjective and Stochastic Optimization based on Parametric Optimization*. Akademie-Verlag, 1985.

[51] P. Gupta, R. Janardan, M. Smid, and B. Dasgupta. The rectangle enclosure and point-dominance problems revisited. *Int. J. Comput. Geom. Appl.*, 5:437–455, 1997.

[52] W. Habenicht. Quad trees, a datatructure for discrete vector optimization problems. *Lecture Notes in Economics and Mathematical Systems*, 209:136–145, 1983.

[53] J. K. Hale. *Theory of Functional Differential Equations*. Springer Verlag, New York, 1977.

[54] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, 1992.

[55] H. Hemami, F.C. Weimer, C.S. Robinson, W.C. Stockwell, and V.S. Cvetkovic. Biped stability considerations with vestibular models. *IEEE Trans. Automatic Control*, 23:1074–1079, 1978.

[56] M. E. Henderson. Multiple parameter continuation: Computing implicitly defined k-manifolds. *International Journal of Bifurcation and Chaos*, 12:451–476, 2002.

[57] T. Hestermeyer and O. Oberschelp. Selbstoptimierende Fahrzeugregelung - Verhaltensbasierte Adaption. In *Intelligente mechatronische Systeme*, volume 122 of *HNI-Verlagsschriftenreihe*. Heinz Nixdorf Institut, 2003.

[58] C. Hillermeier. *Nonlinear Multiobjective Optimization - A Generalized Homotopy Approach*. Birkhäuser, 2001.

[59] D. M. Himmelblau. *Applied Nonlinear Programming*. McGraw Hill Book Company, 1972.

[60] M. W. Hirsch and S. Smale. On algorithms for solving $f(x) = 0$. *Communications on Pure and Applied Mathematics*, 32:281–312, 1979.

[61] A. Hohmann. An adaptive continuation method for implicitly defined manifolds. Konrad-Zuse-Zentrum Berlin, Technical Report, 1991.

[62] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer, 1993.

[63] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer, 1996.

[64] H. Hsu. Global analysis by cell mapping. *Int. J. Bif. Chaos*, 2:727–771, 1992.

[65] J. Jahn. *Mathematical Vector Optimization in Partially Ordered Linear Spaces*. Verlag Peter Lang GmbH, Frankfurt am Main, 1986.

[66] O. Junge. *Mengenorientierte Methoden zur numerischen Analyse dynamischer Systeme*. PhD thesis, University of Paderborn, 1999.

[67] A. Jüschke and J. Jahn. A bicriterial optimization problem of antenna design. *Comp. Opt. Appl.*, 7:261–276, 1997.

[68] W. E. Karush. *Minima of functions of several variables with inequalities as side conditions*. PhD thesis, Master's Dissertation, University of Chicago, 1939.

[69] B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer, 1996.

[70] R. B. Kellogg, T. Y. Li, and J. Yorke. A constructive proof of the brouwer fixed-point theorem and computational results. *SIAM J. Numer. Anal.*, 13:473–483, 1976.

[71] J. Knowles and D. Corne. On metrics for comparing nondominated sets. In *Congress of Evolutionary Computation (CEC2002)*, pages 711–716, 2002.

[72] V. B. Kolmanovskii and V. R. Nosov. *Stability of Functional Differential Equations*. Academic Press, London, 1986.

[73] P. Kravanja, M. Van Barel, O. Ragos, M.N. Vrahatis, and F. A. Zafirpoulos. ZEAL: A mathematical software package for computing zeros of analytic functions. *Computer Physics Communications*, 124:212–232, 2000.

[74] H. Kuhn and A. Tucker. Nonlinear programming. *Proc. Berkeley Symp. Math. Statist. Probability, (J. Neumann, ed.)*, pages 481–492, 1951.

[75] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. On the convergence and diversity-preservation properties of multi-objective evolutionary algorithms. TIK-Report No. 108, ETH Zürich, 2001.

[76] A. Lotov, V. A. Bushenkov, and G. Kamenev. *Interactive Decision Maps. Approximation and Visualization of Pareto Frontier*. Kluwer Academic Publishers, 2004.

[77] E. M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14:257–276, 1985.

[78] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[79] M. Mitschke. *Dynamik der Kraftfahrzeuge*, volume C: Fahrverhalten. Springer Verlag, 2nd edition, 1990.

[80] J. Moré, B. Garbow, and K. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Soft.*, 7:17–41, 1981.

[81] A. Morgan. A homotopy for solving general polynomial systems that respects m-homogenuous structures. *Appl. Math. Comp.*, 24:101–113, 1987.

[82] S. Mostaghim. *Multi-Objective Evolutionary Algorithms, Data Structures, Convergence and Diversity*. PhD thesis, University of Paderborn, 2004.

[83] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization. In *IEEE 2003 Swarm Intelligence Symposium*, IEEE Press, 2003.

[84] S. Mostaghim, J. Teich, and A. Tyagi. Comparison of data structures for storing pareto-sets in moeas. *Int. J. Comput. Geom. Appl.*, 5:437–455, 2002.

[85] E. Münch. *Mehrgrößenoptimierung - Algorithmusentwicklung an der Spurführung der NPB (Neue Bahntechnik Paderborn).* Diploma Thesis, University of Paderborn, 2004.

[86] Yu. I. Neimark. The structure of the D-partitioning of the space of quasipolynomials and the diagramms of Vishnegradskii and Nyquist. *Dok. Adad. Sci. USSR*, 60:1503–1506, 1948.

[87] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[88] H. Nyquist. Regeneration theory. *Bell Syst. Tech. J.*, 11(1):126–147, 1932.

[89] T. Oktabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimization. In *IEEE Congress on Evolutionary Computation*, IEEE Press, 2003.

[90] V. Pareto. *Cours d'Economie Politique.* Libraire Droz, Genève, 1964 (first edition in 1896).

[91] H.-O. Peitgen and M. Prüfer. The Leray-Schauder continuation method is a constructive element in the numerical study of nonlinear eigenvalue and bifurcation problems. In H.-O. Peitgen and H.O. Walther, editors, *Functional Differential Equations and Approximation of Fixed Points*, volume 730 of *Lecture Notes in Mathematics*, 1979.

[92] H. O. Peitgen, M. Prüfer, and K. Schmitt. Global aspects of the continuous and discrete Newton method: a case study. *Acta. Appl. Math.*, 13:123–202, 1988.

[93] L. S. Pontryagin. On the zeros of some elementary transcendental functions. *AMS Transl.*, 1:545–552, 1955.

[94] A. Pottharst, K. Baptist, O. Schütze, J. Böcker, N. Fröhlecke, and M. Dellnitz. Operating point assignment of a linear motor driven vehicle using multiobjective optimization methods. *Proceedings of the 11th International Conference EPE-PEMC 2004, Riga, Latvia.*, 2004.

[95] F. P. Preparata and M. I. Shamos. *Computational Geometry - An Introduction.* Springer Verlag, 1988.

[96] J. Rakowska, R. T. Haftka, and L. T. Watson. Tracing the efficient curve for multi-objective control-structure optimization. *Computing Systems in Engineering*, 2(6):461–471, 199.

[97] W. Rheinboldt. On the computation of multi-dimensional solution manifolds of parametrized equations. *Numer. Math.*, 53:165–181, 1988.

[98] W. C. Rheinboldt. *Numerical Analysis of Parametrized Nonlinear Equations.* Wiley, 1986.

[99] W. C. Rheinboldt. On a moving frame algorithm and the triangulation of equilibrium manifolds. In T. Küpper, R. Seydel, and H. Troger, editors, *Bifurcation: Analysis, Algorithms, Applications*, Birkhäuser, Basel, 1987.

[100] G. Rudolph. Finite Markov chain results in evolutionary computation: A tour d'horizon. *Fundamenta Informaticae*, 35:67–89, 1998.

[101] G. Rudolph. On a Multi-Objective Evolutionary Algorithm and Its Convergence to the Pareto Set. In *5th IEEE Conference on Evolutionary Computation*, pages 511–516, 1998.

[102] G. Rudolph and A. Agapie. On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In *Congress on Evolutionary Computation (CEC2000)*, pages 1010–1016, 2000.

[103] S. Schäffler, R. Schultz, and K. Weinzierl. A stochastic method for the solution of unconstrained vector optimization problems. *J. Opt. Th. Appl.*, 114(1):209–222, 2002.

[104] K. Schittkowski. EASY-OPT: An interactive optimization system with automatic differentiation - user's guide. Department of Mathematics, University of Bayreuth, Technical Report, 1999.

[105] K. Schittkowski. NLPJOB version 2.0: A Fortran code for multicriteria optimization - user's guide. Department of Mathematics, University of Bayreuth, Technical Report, 2003.

[106] O. Schütze. *Zur globalen Lösung nichlinearer Gleichungssysteme mit Hilfe von Unterteilungsalgorithmen.* Diploma Thesis, University of Bayreuth, 1999.

[107] O. Schütze. A new data structure for the nondominance problem in multi-objective optimization. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization (EMO 03)*, volume 2 of *Springer*, 2003.

[108] O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. Covering Pareto sets by multilevel evolutionary subdivision techniques. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, 2003.

[109] S. Sertl. *Parallele Algorithmen zur Analyse dynamischer Systeme.* Diploma Thesis, University of Bayreuth, 1998.

[110] S. Sertl. *Parallele Algorithmen zur globalen Optimierung.* PhD Thesis, University of Paderborn (unpublished), 2004.

[111] S. Sertl and M. Dellnitz. Global optimization using a dynamical systems approach. To appear in *Journal of Global Optimization*, 2005.

[112] M. Shub. *Global Stability of Dynamical Systems.* Springer, 1987.

[113] G. Stépán. *Retarded Synamical Systems: Stability and Characteristic Functions.* Longman Scientific & Technical, 1989.

[114] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation, and Applications*. John Wiley & Sons, Inc., 1986.

[115] R. E. Steuer, editor. *Multiple Criteria Decision Making and Risk Analysis Using Microcomputers*, chapter The Tchebycheff Procedure of Interactive Multiple Objective Programming, pages 235–249. Springer, Berlin, 1989.

[116] M. Sun and R. E. Steuer. InterQuad: An interactive quad tree based procedure for solving the discrete alternative multiple criteria problem. *European Journal of Operational Research*, 89:462–472, 1996.

[117] M. Sun and R.E. Steuer. Quad-trees and linear lists for identifying nondominated criterion vectors. *INFORMS J. Comput.*, 4:367–375, 1996.

[118] H. Trinkaus and T. Hanne. knowCube: a visual and interactive support for multicriteria decision making. *Computers and Operations Research*, 32:1289–1309, 2005.

[119] J. Verschelde and A: Haegemans. Homotopies for solving polynomial systems within a bounded domain. *Theoretical Comp. Sci. A.*, 133(3):165–185, 1994.

[120] A. Wolf. *Zur numerischen Berechnung des Abbildungsgrades*. Diploma Thesis, University of Paderborn, 2002.

[121] X. Ying and N. Katz. A reliable argument principle algorithm to find the number of zeros of an analytic function in a bounded domain. *Numerische Mathematik*, 53:143–163, 1988.

[122] 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.

[123] Q. Zheng and M. Dellnitz. Schwingungen eines Ringoszillators – eine numerische Behandlung unter Berücksichtigung der Symmetrie. *ZAMM*, 70:T135 – T138, 1990.

[124] E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne. *Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001.* Springer, 2001.

[125] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. In *Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, 2002.

[126] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and and the strength Pareto approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, 1999.