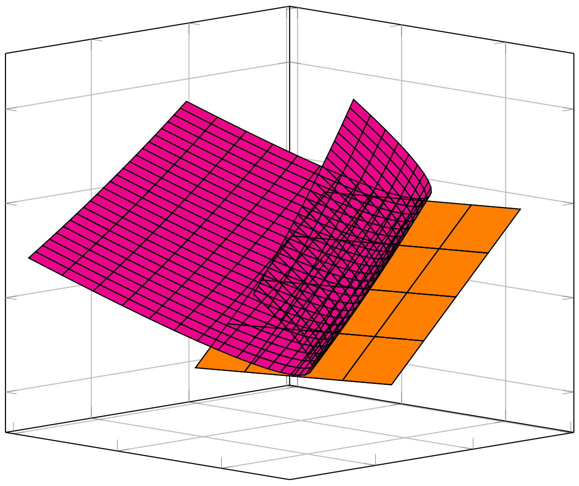
Universität Bremen

Use of Parametric Sensitivities for Multi-Objective Optimisation



Dissertation

zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

– Dr. rer. nat. –

der Fakultät für Mathematik und Informatik

an der Universität Bremen vorgelegt von

Arne Berger

Datum des Promotionskolloquiums: 27. Juni 2022

1. Gutachter: Prof. Dr. Christof Büskens, Universität Bremen
2. Gutachterin: Prof. Dr. Kathrin Flaßkamp, Universität des Saarlandes

Bremen 2021

Acknowledgements

This thesis and the process of learning, research and experimenting would not have been possible without the constant support, encouragement and assistance of people from both the academic and social spheres, and I am grateful to all of them.

First and foremost, my special thanks go to my wife, Katharina Berger. She supported me all the years in my progress and my struggles throughout this work. This support enabled me to complete the present work despite all difficulties and doubts.
I also want to thank my parents, who always supported me on my path to learn and study mathematics.

Furthermore, I thank my supervisor Prof. Dr. Christof Büskens, who guided me through my academic path and always had an open ear for my questions. He offered me an environment in which I could develop ideas far beyond purely academic work so that I could grow through a variety of challenges during my time as a doctoral student.

I am grateful for the many talks and ideas I discussed with my unofficial co-supervisor Dr.-Ing. Mitja Echim and my former colleague Dr. Sören Geffken. I would also like to thank my colleagues Dr.-Ing. Francesca Jung and Wiebke Bergmann for their support in my projects and the fruitful discussions on teamwork and agile methods.

For their opinions and corrections on my work I thank my colleagues Dr. Sylvain Roy, Dr. Matthias Knauer, Marcel Jacobse, Maria Höffmann and Marek Wiesner.

I would like to thank Prof. Dr. Jörg Fliege for providing me with a place as a visiting researcher for three months in 2019 and for helping me with ideas and opinions on my work on the MOSQP algorithm.

Lastly, I want to thank all my former and present colleagues in the Working Group Optimisation and Optimal Control who created a joyful and inspiring environment for me to work in.

Abstract

When controlling automated systems, the combination of several different objectives becomes increasingly important. These objectives represent goals of the process like driving as close as possible to a final position or minimising the energy consumption as well as necessary numerical goals like the regularisation of controls. Integrating these objectives into an automated system can become complicated when the applied algorithms behave unintuitively or take too long to compute.

This thesis aims at investigating how parametric sensitivities can be utilised to provide the developer of automated systems with a better insight into the structure of the underlying multi-objective problem. For this, the Pascoletti-Serafini approach is used. In this approach, the Lagrange multiplier represents a linearisation of the Pareto front at a computed sample. Based on this, the Pareto front is interpolated locally around a computed sample. In addition, the parametric sensitivities of the Lagrange multiplier are used to extend this approach to a quadratic approximation. A third approach is formulated by using the parametric sensitivities of the variables to create an approximation in variable space. By applying the objective function to the approximated variables, another approximation of the Pareto front is computed.
Next, the local approximations are used to define a continuous approximation of the whole Pareto front by defining a polynomial interpolation using the samples and derivative information. This interpolation is done for all three variants of approximations in a bi-objective setting. The limitations of this method, mainly rooted in the locality of parametric sensitivity information, are discussed afterwards.

Next, the MOSQP algorithm is introduced as a method to investigate multi-objective problems with less necessity for a-priori knowledge than methods like the Pascoletti-Serafini scalarisation. The algorithm is improved by replacing the internal cleanup strategy between iterations by the strategy from the heuristic NSGA-II algorithm. In this way, fewer points are deleted from the intermediate set in early iterations, which improves the overall spread. The improvement is benchmarked using performance profiles.

Finally, the introduced methods are applied to a bi-objective example of autonomous ship control from the research project GALILEOnautic 2 using the dynamical model of a real ship. The methods are evaluated for their usability within the development process of a trajectory optimisation module. Particular attention is paid to how much information the methods provide to the developer, especially when reducing the number of samples, and how intuitively they behave.

Zusammenfassung

Bei der Steuerung automatisierter Systeme wird die Kombination mehrerer verschiedener Ziele immer wichtiger. Bei diesen Zielen handelt es sich sowohl um Ziele aus dem Prozess, wie z. B. das Anfahren einer Endposition oder ein möglichst geringer Energieverbrauch, als auch um notwendige numerische Ziele, wie z. B. die Regularisierung von Steuerungen. Die Integration dieser Ziele in ein automatisiertes System kann kompliziert werden, wenn die angewandten Algorithmen sich unintuitiv verhalten oder zu lange für die Berechnung benötigen.

In dieser Arbeit wird untersucht, wie parametrische Sensitivitäten genutzt werden können, um dem Entwickler automatisierter Systeme einen besseren Einblick in die Struktur des zugrunde liegenden multikriteriellen Problems zu geben. Hierfür wird der Pascoletti-Serafini-Ansatz verwendet. Hierbei stellt der Lagrange-Multiplikator eine Linearisierung der Pareto-Front an einem berechneten Sample. Hiermit wird die Pareto-Front lokal um ein berechnetes Sample herum interpoliert. Darüber hinaus werden die parametrischen Sensitivitäten des Lagrange-Multiplikators genutzt, um diesen Ansatz auf eine quadratische Approximation zu erweitern. Ein dritter Ansatz wird formuliert, indem die parametrischen Sensitivitäten der Variablen verwendet werden, um eine Approximation im Variablenraum zu erstellen. Durch Anwendung der Zielfunktion auf die approximierten Variablen wird eine weitere Annäherung an die Pareto-Front berechnet.
Anschließend werden die lokalen Approximationen verwendet, um eine kontinuierliche Approximation der gesamten Pareto-Front zu definieren, indem eine Polynominterpolation unter Verwendung der Samples und der Ableitungsinformationen definiert wird. Diese Interpolation wird für alle drei Varianten der Approximationen in einer bi-objektiven Situation durchgeführt. Die Grenzen dieser Methode, die hauptsächlich in der Lokalität der Informationen über die parametrische Sensitivität begründet sind, werden anschließend erörtert.

Als nächstes wird der MOSQP-Algorithmus als eine Methode zur Untersuchung von Mehrzielproblemen vorgestellt, die weniger a-priori-Wissen erfordert als Methoden wie die Pascoletti-Serafini-Skalarisierung. Der Algorithmus wird verbessert, indem die interne Bereinigungsstrategie zwischen den Iterationen durch die Strategie des heuristischen NSGA-II-Algorithmus ersetzt wird. Auf diese Weise werden in den ersten Iterationen weniger Punkte aus der vorläufigen Menge gelöscht, was die Gesamtverteilung verbessert. Die Verbesserung wird mit Hilfe von Performance-Profilen überprüft.

Abschließend werden die vorgestellten Methoden auf ein Beispiel mit zwei Zielfunktionen aus der autonomen Schiffssteuerung aus dem Forschungsprojekt GALILEOnautic 2 unter Verwendung des dynamischen Modells eines realen Schiffes angewendet. Die Methoden werden auf ihre Nützlichkeit im Rahmen des Entwicklungsprozesses eines Moduls zur Trajektorienoptimierung untersucht. Dabei wird berücksichtigt, wie viele Informationen die Methoden dem Entwickler zur Verfügung stellen, insbesondere bei der Reduzierung der Anzahl von Samples, und wie intuitiv sie sich verhalten.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Algorithms that continuously find and apply the best compromise for a situation are a cornerstone of the automated systems of the future. To develop these algorithms, the developer has to understand the underlying system and the compromises it creates. This thesis aims at improving the existing tools that guide the developer in this understanding. Based on multi-objective optimisation and parametric sensitivity analysis, new ways are described to gather as much information about the underlying system as possible, thus creating a fast and safe development process for future systems.

Finding compromises is fundamental. Every day we make decisions and take several goals into account. We sometimes think through some decisions consciously: "Should I buy the expensive and performant computer or instead the cheaper but slower one?", but we also make others unconsciously: "I want to drive through this curve as fast as possible but still feel safe". Humans make and adjust these decisions instinctively by selecting one of the possible compromises. However, expressing them in measurable equations such that a computer system could automate the decision making process is hard. Even though individual goals like "process time" and "rotational forces" are often easily computable, it is not easy to express the overall optimisation criterium. It is not even clear what optimal means in the situation with several conflicting goals.

How these problems are tackled and how the process of doing so can be improved is the topic of this thesis.

1.1 Goal

This thesis aims at improving multi-objective optimisation tools that are used in the development process of automated systems. In this context an improvement means that the developer gets more information from the existing process or that the tool becomes faster. Both cases aim at making the overall development process more efficient.

The theory of multi-objective optimisation introduces the term "Pareto-optimality" to define what is considered an optimal compromise. These compromises are computed through scalarisa-

tion techniques that provide a finite set of possible best compromises [91]. The process of deciding which of these compromises is the most favourable is referred to as decision-making [20, 59]. In real-world applications, this process is often long and sometimes frustrating because the scalarisation techniques need much adjustment of hyperparameters before they provide a suitable result. Two possibilities exist to overcome this problem: Provide the developer with more information, such that they understand the system better and thus find appropriate hyperparameters more quickly, or discover algorithms that are free of these hyperparameters. Both ways are addressed and improved in this thesis.
The first way is addressed by connecting parametric sensitivity analysis with multi-objective optimisation such that it becomes possible to create continuous approximations of the Pareto front. The goal here is to get more information on the shape of the Pareto front from one computed sample, thus providing new insights and reducing the number of samples that need to be computed.
The second possibility is addressed by the MOSQP algorithm that was developed by Fliege and Vaz [46] to provide a derivative-based algorithm that can find a whole set of Pareto-optimal points without the need of an additional hyperparameter. Here the goal of this thesis is to improve the existing algorithm such that the information provided by the computed set of points is enhanced.

To evaluate how the described tools perform when used in the development process of automated systems, they are applied to an example from autonomous ship manoeuvring in the context of the research project GALILEOnautic 2 [52]. The goal is to show how multi-objective optimisation is helpful in this application in general and how the proposed methods can be used in particular.

1.2 Outline

This thesis consists of the following chapters: The current **Chapter 1** contains the introduction, goal, outline and a brief overview of the scientific contribution.
Chapter 2 gives an introduction to the mathematical fundamentals of nonlinear single-objective and multi-objective optimisation. It also describes parametric sensitivity analysis and the fundamentals of the SQP-method and the NLP-Solver WORHP. For the numerical solution of multi-objective optimisation problems, three scalarisation methods are described.
Chapter 3 uses parametric sensitivity analysis to build continuous approximations of the Pareto front for multi-objective optimisation problems. The method is described and afterwards investigated by applying it to several academic examples.
In **Chapter 4** the MOSQP algorithm is described. The chapter then highlights a weakness of the algorithm and describes how to overcome it. The impact of the proposed change is also investigated in this chapter.
Chapter 5 explains how multi-objective optimisation can be utilised in the development of a nonlinear model predictive controller for autonomous ship manoeuvring. The chapter applies the methods described in this thesis to an example from the project GALILEOnautic 2 and investigates how these techniques can help in the development process.
Chapter 6 provides some thoughts on the implementation of the described methods within one interface of the solver WORHP. It gives some implementation details with the main focus on ideas on how to parallelise the computation.
Finally, **Chapter 7** gives a brief conclusion and outline for future work.

1.3 Contribution of this Thesis

Using Parametric Sensitivity Analysis Information for better Pareto Front Approximations

Chapter 3 discusses how parametric sensitivity analysis is utilised to build continuous Pareto front approximations of bi-objective problems. In Section 3.1 the local approximations based on the Lagrange multipliers [32, 61] are extended in Section 3.1.1 with the second derivatives. Also, Section 3.1.2 adds approximations for the variables leading to another approximation of the Pareto front.

Afterwards, Section 3.2 proposes a technique to compute an approximation that is continuous on the connected parts of the sampled Pareto front. This technique is also applied directly in the objective space as well as for the variables. An introduction to this topic is also provided in Berger, Knauer and Büskens [7] and Berger and Büskens [5].
The limitations of the proposed methods are discussed in Section 3.3 before Section 3.4.1 shows how the local approximation is extended for more than two objective functions.

Advances in Multi-Objective Sequential Quadratic Programming

Section 4.2 discusses the impact of the cleanup strategy within the MOSQP algorithm (Algorithm 4.13). To the knowledge of the author, the usage of the crowding distance 4.10 as the basis for performing a cleanup between iterations in the MOSQP algorithm is a new idea first applied to the MOSQP algorithm in this thesis.
Also, Section 4.3 discusses the new approach of utilising the NLP solver WORHP for the iterations within the MOSQP algorithm that helps in reusing some of the techniques within this solver like building the necessary quadratic problems or performing a line-search for the iteration step size.
Section 4.5 discusses strategies for building the initial set of points given to the algorithm. For this, Strategy 4.23, which computes points between the individual minima instead of the box-bounds, is newly introduced. Also Strategies 4.24 and 4.25, which refine these points between the extrema, are added to the already known strategies for the algorithm.
Finally, the implementation considerations in Section 6.2.2, that mainly discuss how the spread phase of the MOSQP Algorithm can be parallelised, add essential details to the applicability of the algorithm in real-world examples.

Application of Multi-Objective Optimisation within Nonlinear Model Predictive Control of Autonomous Ships

In Chapter 5, it is shown how multi-objective optimisation and especially the new methods described in this thesis can be utilised within the development process of a nonlinear model predictive control (nMPC) module, for instance, within the the project GALILEOnautic 2 [52]. The focus is on creating insight into the problem that guides the developer towards an optimal controller setting. Section 5.2.1 discusses the different scalarisation methods.

Afterwards, Section 5.2.2 shows how the local approximation provides new information within the context of developing a nonlinear model predictive controller for autonomous ship manoeuvring. Section 5.2.2 discusses the application of the global continuous approximation to the same context. Both sections focus on generating insight into the discussed applications that is only possible with the proposed methods and reducing the number of samples needed to reduce computation time.

Chapter 2

Mathematical Fundamentals of Nonlinear Optimisation

Contents

2.1 Fundamentals of Nonlinear Constrained Single-Objective Optimisation

Optimisation is the theory of finding a solution for a given problem which is in some regard better than all other solutions. In unconstrained optimisation the possible solutions are usually points in some vector space which are compared by a problem specific function from this vector space into the real numbers, the so-called objective function. In constrained optimisation, the problem usually involves constraints which must be satisfied by the possible solutions which are then called feasible. In the following a formal mathematical definition of this task is given.

Definition 2.1 (Standard nonlinear optimisation problem)
Let the dimensions $0 < N_x \in \mathbb{N}$, $0 \leq N_g \in \mathbb{N}$ and $0 \leq N_h \in \mathbb{N}$ be given. For the continuous functions $f : \mathbb{R}^{N_x} \to \mathbb{R}$, $g : \mathbb{R}^{N_x} \to \mathbb{R}^{Ng}$ and $h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_h}$ with $x \in \mathbb{R}^{N_x}$ the **standard nonlinear optimisation problem (NLP)** is defined by

$$\begin{aligned} \min_{x\in\mathbb{R}^{N_x}} \quad & f(x), \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \dots, N_g, \\ & h_j(x) = 0, \quad j = 1, \dots, N_h. \end{aligned} \tag{NLP}$$

In this context the function $f(x)$ is called **objective function**, $g(x)$ describes the **inequality constraints** and $h(x)$ the **equality constraints**.

Definition 2.1 formulates the general task of nonlinear continuous constrained optimisation. However, the term minimum (min) has not yet been formally defined, which is done in what follows. For this purpose, the set of feasible points is to be defined first.

Definition 2.2
The set of feasible points for a problem of the form (NLP) is defined by

$$\Sigma = \left\{ x \in \mathbb{R}^{N_x} \mid g(x) \leq 0, h(x) = 0 \right\}. \tag{2.1}$$

Now it is possible to give a formal definition of optimality and the term minimum.

Definition 2.3 (Optimality)
For (NLP) a point $x^* \in \Sigma$ is called:

- **Globally optimal** if and only if $f(x^*) \leq f(x)$ for all $x \in \Sigma$.
- **Locally optimal** if and only if there exists a neighbourhood $S_{x^*} \subset \mathbb{R}^{N_x}$ around x^* and $f(x^*) \leq f(x)$ for all $x \in S_{x^*} \cap \Sigma$.

If, in the equations above, the relation "$<$" holds for all $x \neq x^*$ the point x^* is called **strictly (globally / locally) optimal**. Usually x^* is also called a (global / local) minimum.

Since Σ usually holds an infinite number of points, it is in general not possible to test a point directly for optimality by comparing it to all other points. The next section deals with formulating conditions that can be used to prove local optimality.

Note 2.4
In the context of convex [12] or linear [30] optimisation, every local minimum is also a global minimum. This characteristic is an important advantage when solving these types of problems, since derivative-based optimisation solvers only find local minima. However, this thesis focusses on general nonlinear optimisation problems for which no general statement about locality of an optimum can be made.

Note 2.5
Optimality is often identified with minimality. Since the search for a maximum of f is equivalent to the minimisation of $-f$, only minimisation tasks are discussed hereafter.

2.1.1 Optimality Conditions

This section discusses necessary and sufficient conditions for locally optimal points x^* of the problem (NLP). The focus is on those conditions which are used in the following chapters. A more complete discussion of the subject can be found in several books, see for instance the books by Nocdeal and Wright [95], Geiger and Kanzow [53] or Fletcher [41].

First of all, the optimality conditions from the unconstrained optimisation should be recalled. In the case, that in Definition 2.1, $N_g = 0$ and $N_h = 0$ and the function f is continuously differentiable, a necessary condition for a locally optimal point x^* is given by

$$\nabla_x f(x^*) = 0. \tag{2.2}$$

If, in addition, f is twice continuously differentiable and

$$\nabla^2_{xx} f(x^*) > 0, \tag{2.3}$$

then it follows directly that x^* is locally optimal and one refers to (2.2) together with (2.3) as sufficient second-order conditions (see [41]).

The goal is now to find similar conditions for constrained problems of type (NLP). For this, the Lagrange function is usually defined, which represents a first relationship between constrained and unconstrained optimisation.

Definition 2.6 (Lagrange function)
For the vectors $\lambda \in \mathbb{R}^{N_g}$ and $\mu \in \mathbb{R}^{N_h}$, the function

$$L(x, \lambda, \mu) \coloneqq f(x) + \lambda^\top g(x) + \mu^\top h(x), \tag{2.4}$$

is defined as the **Lagrange function**, where λ and μ are called the **Lagrange multiplicators**.

Both Karush, as well as Kuhn and Tucker, independently introduced the following conditions, commonly referred to as KKT conditions. Karush introduced the KKT conditions in his unpublished master thesis (1939), which is reviewed in [81] and [71], while Kuhn and Tucker, independently of Karush's work, published their work (1951) originally in [82] which was republished several times. The most recent version (2014) is published in [83].

Definition 2.7 (KKT conditions)
Given the dimensions $0 < N_x \in \mathbb{N}$, $0 \leq N_g \in \mathbb{N}$ and $0 \leq N_h \in \mathbb{N}$, let the functions $f : \mathbb{R}^{N_x} \to \mathbb{R}$, $g : \mathbb{R}^{N_x} \to \mathbb{R}^{N_g}$ and $h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_h}$ be continuously differentiable. Then a point $(x^*, \lambda^*, \mu^*) \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_g} \times \mathbb{R}^{N_h}$ satisfies the Karush-Kuhn-Tucker conditions if and only if

$$\nabla_x L(x^*, \lambda^*, \mu^*) = 0, \tag{2.5}$$
$$g_j(x^*) \leq 0, \quad i = 1, \dots, N_g, \tag{2.6}$$
$$h_i(x^*) = 0, \quad j = 1, \dots, N_h, \tag{2.7}$$
$$\lambda_i^* g_i(x^*) = 0, \quad i = 1, \dots, N_g, \tag{2.8}$$
$$\lambda_i^* \geq 0, \quad i = 1, \dots, N_g. \tag{2.9}$$

In this case, the point (x^*, λ^*, μ^*) is called a **KKT-Point**. Equation (2.5) is called the optimality condition, (2.6) and (2.7) the feasibility conditions and (2.8) the complementary condition.

The KKT conditions form the basis for most of the optimality conditions. However, they are usually extended by requirements on the constraint functions in the point x^*. One such condition which seems quite natural is the linear independence constraint qualification which is given below. For further conditions and detailed explanations, see for example [53, 95]. But first the so-called active set is defined.

Definition 2.8 (Active Set)
For a point $x \in \mathbb{R}^{N_x}$ such that $g_i(x) \leq 0$, $i = 1, \dots, N_g$, the **active set** of x is defined as the set of indices

$$\mathcal{A}(x) \coloneqq \{i \mid g_i(x) = 0,\ i \in \{1, \dots, N_g\}\}. \tag{2.10}$$

In addition, for $i \in \{1, \dots, N_g\}$ a constraint $g_i(x)$ is called active if $i \in \mathcal{A}(x)$.

Definition 2.9 (Constraint qualification)
For the constraint functions $g(x)$ and $h(x)$ given as in Definition 2.7 a point $x^* \in \Sigma$ satisfies the **linear independence constraint qualification (LICQ)**) if the gradients of all active constraints

$$\nabla_x g_i(x^*),\ i \in \mathcal{A}(x^*) \text{ and } \nabla_x h_j(x^*),\ j = 1 \dots, N_h, \tag{2.11}$$

are pairwise linearly independent.

Together with the KKT conditions (2.5) – (2.9) it is now possible to state necessary conditions for a local optimum.

Theorem 2.10 (First order necessary conditions under LICQ)
If x^* is a local minimum of (NLP) and satisfies the LICQ conditions in Definition 2.9, then there exist uniquely determined multipliers λ^* and μ^* such that (x^*, λ^*, μ^*) satisfy the KKT conditions (2.5) - (2.9).

Proof: See [41]. □

Theorem 2.10 gives only necessary conditions which are also satisfied by saddle points and maxima for the problem (NLP). For this reason, further conditions are defined analogously to the unconstrained optimisation, which, together with the necessary conditions, form sufficient conditions. For this, the Lagrange function is used instead of the objective function. In contrast to unconstrained optimisation, the conditions do not have to apply everywhere but only along feasible decent directions. These directions are given by the *critical cone* (see [53]). Since the critical cone in its original form is difficult to handle, one usually uses the linearised form.

Definition 2.11 (Linearised critical cone)
Let the functions f, g and h be given as in Definition 2.7. For $x \in \Sigma$, $\lambda \in \mathbb{R}^{N_g}$ the set

$$\begin{aligned} C(x,\lambda) := \big\{ d \in \mathbb{R}^{N_x} |\ & \nabla_x g_i(x)^\top d \leq 0 \text{ for } i \in \mathcal{A}(x)\ \lambda = 0, \\ & \nabla_x g_i(x)^\top d = 0 \text{ for } i \in \mathcal{A}(x)\ \lambda > 0, \\ & \nabla_x h_j(x)^\top d = 0 \text{ for } j = 1, \dots, N_h \big\}, \end{aligned} \tag{2.12}$$

is called **linearised critical cone** of x.

With the linearised ciritical cone (2.12) it is now possible to define sufficient conditions which are very similar to the ones for unconstrained optimisation.

Theorem 2.12 (Sufficient conditions)
Let the functions f, g and h be given as in Definition 2.7 and additionally be twice continuous differentiable. Also, let the point (x^*, λ^*, μ^*) satisfy all conditions of Theorem 2.10. If, in addition, the condition

$$d^\top \nabla^2_{xx} L(x^*, \lambda^*, \mu^*) d > 0 \ \text{ for all } d \in C(x^*, \lambda^*), \tag{2.13}$$

is satisfied, then (x^*, λ^*, μ^*) is strictly locally optimal for Problem (NLP).

Proof: See [41]. □

The complementarity condition (2.8) from Definition 2.7 allows in its general form the degenerate case that

$$\lambda_i = 0, \text{ for all } i \in \mathcal{A}(x^*). \tag{2.14}$$

The constraint g_i is therefore active and at the same time the corresponding Lagrange multiplier λ_i is 0. In Section 2.2, it will be shown that λ_i represents the sensitivity of the objective function with regard to g_i. The case in (2.14) would mean that g_i, despite being active, would have no influence on the objective function. In order to exclude this case, the *strict sufficient optimality conditions* can be stated.

Lemma 2.13 (Strict sufficient optimality conditions)
If condition (2.9) is complemented by the requirement that

$$\lambda_i > 0 \ \text{ for all } i \in \mathcal{A}(x^*), \tag{2.15}$$

applies, then the critical cone is simplified to the kernel of the matrix formed by all active inequality constraints and the equality constraints

$$C(x^*, \lambda^*) = \ker\left(\nabla_x g_i(x^*), \nabla_x h_j(x^*)\right), \quad i \in \mathcal{A}(x^*),\ j = 1, \dots, N_h.$$

In this case, the requirements of Theorem 2.12 together with (2.15) are referred to as **strict sufficient optimality conditions**. See for example [18].

Proof: See [41]. □

Note 2.14
Numerically it is possible to determine the kernel $\ker\left(\nabla_x g_i(x^*), \nabla_x h_j(x^*)\right)$ via, for instance, a QR-decomposition. In practical use, however, this is usually not explicitly necessary.

2.1.2 Numerical Solution of Nonlinear Optimisation Problems with Sequential Quadratic Programming

Section 2.1.1 describes the theoretical fundamentals of nonlinear optimisation and explains necessary and sufficient criteria for a point $x^* \in \mathbb{R}^{N_x}$ to be optimal for Problem (NLP). This section gives an introduction to the numerical solution of (NLP) via Sequential Quadratic Programming (SQP).
More specifically, this section shows how an iterative solver can improve a given point $x^{[k]}$ iteratively such that the iteration over $x^{[k]}$ converges for $k \to \infty$ to a local minimum of problem (NLP). For this, first, the general idea of the SQP method is explained. Afterwards, the formal local convergence is shown, and afterwards, an overview of globalisation strategies is given.

In order to show the core idea of the SQP-Algorithm, the problem (NLP) is reduced to consider only equality constraints. This reformulation yields the problem

$$\begin{aligned} \min_x \quad & f(x), \\ \text{subject to} \quad & h_j(x) = 0, \quad j = 1, \dots, N_h, \end{aligned} \tag{2.16}$$

with the corresponding Lagrange function

$$L(x, \mu) = f(x) + \mu^\top h(x). \tag{2.17}$$

The KKT-conditions (2.5) – (2.7) for Problem (2.16) are then represented by

$$\Phi(x, \mu) := \begin{pmatrix} \nabla_x L(x, \mu) \\ h(x) \end{pmatrix} = 0. \tag{2.18}$$

The idea is now to apply Newton's method in order to find the zeros of $\Phi(x, \mu)$ and therefore KKT-points (x^*, μ^*) of problem (2.16). For the iteration rule

$$\begin{aligned} x^{[k+1]} &:= x^{[k]} + d^{[k]}, \\ \mu^{[k+1]} &:= \mu^{[k]} + \Delta\mu^{[k]}, \end{aligned} \tag{2.19}$$

the required steps $(d^{[k]}, \Delta\mu^{[k]})$ can be found by applying Newton's method to (2.18) which leads to the linear equation system

$$\Phi'(x^{[k]}, \mu^{[k]}) \begin{pmatrix} d^{[k]} \\ \Delta\mu^{[k]} \end{pmatrix} = -\Phi(x^{[k]}, \mu^{[k]}), \tag{2.20}$$

where

$$\Phi'(x,\mu) = \begin{pmatrix} \nabla^2_{xx} L(x,\mu) & \nabla_x h(x)^\top \\ \nabla_x h(x) & 0 \end{pmatrix}. \tag{2.21}$$

By combining Equation (2.20) and (2.21), the following is obtained:

$$\begin{aligned} \nabla^2_{xx} L(x^{[k]},\mu^{[k]})d^{[k]} + \nabla_x h(x^{[k]})^\top \Delta\mu^{[k]} &= -\nabla_x L(x^{[k]},\mu^{[k]}), \\ \nabla_x h(x^{[k]})d^{[k]} &= -h(x^{[k]}). \end{aligned} \tag{2.22}$$

This can be reformulated, using the relationship $\Delta\mu^{[k]} = \mu^{[k+1]} - \mu^{[k]}$ from Equation (2.19), into

$$\begin{aligned} \nabla^2_{xx} L(x^{[k]},\mu^{[k]})d^{[k]} + \nabla_x h(x^{[k]})^\top \mu^{[k+1]} &= -\nabla_x f(x^{[k]}), \\ \nabla_x h(x^{[k]})d^{[k]} &= -h(x^{[k]}). \end{aligned} \tag{2.23}$$

Thus, $\mu^{[k+1]}$ can be directly calculated. For the iterates of the variable x, however, only a so-called search direction $d^{[k]}$ is determined. The key idea of the SQP-method is now given by the fact that Equation (2.23) also represents the necessary optimality conditions for the quadratic problem

$$\begin{aligned} \min_{d\in\mathbb{R}^{N_x}} \quad & \tfrac{1}{2} d^\top \nabla^2_{xx} L(x^{[k]},\mu^{[k]})d + \nabla_x f(x^{[k]})^\top d, \\ \text{subject to} \quad & h(x^{[k]}) + \nabla_x h(x^{[k]})d = 0. \end{aligned} \tag{2.24}$$

The correlation between the equations (2.23) and (2.24) shows that the computation of a new direction for Newton's method is equivalent to solving a quadratic problem. Thus, since there exist very efficient solvers for problems like (2.24) [54,55,58] it is possible to solve a general (NLP) by sequentially solving quadratic problems. This method is then called Sequential Quadratic Programming (SQP).

Note 2.15
So far, only equality constraints were taken into account. If also inequality constraints shall be considered, they need to be integrated into Equation (2.24) which leads to

$$\begin{aligned} \min_{d\in\mathbb{R}^{N_x}} \quad & \tfrac{1}{2} d^\top \nabla^2_{xx} L(x^{[k]},\lambda^{[k]},\mu^{[k]})d + \nabla_x f(x^{[k]})^\top d, \\ \text{subject to} \quad & g(x^{[k]}) + \nabla_x g(x^{[k]})d \le 0, \\ & h(x^{[k]}) + \nabla_x h(x^{[k]})d = 0. \end{aligned} \tag{2.25}$$

The new value for the dual variables of the inequality constraints $\lambda^{[k+1]}$ are computed analogously to $\mu^{[k+1]}$ directly from the solution of problem (2.25).

Note 2.16
There is no guarantee that there will always be a unique solution to the subproblem (2.25). Thus, if there exist multiple KKT-points of the subproblem, Geiger and Kanzow [53] demand that the KKT-point $\left(x^{[k+1]},\lambda^{[k+1]},\mu^{[k+1]}\right)$ which has the minimal distance to $\left(x^{[k]},\lambda^{[k]},\mu^{[k]}\right)$ is chosen.

So far, this section explains the basic idea behind the SQP method. Next, the convergence of the algorithm is formally investigated in Theorem 2.17.

Theorem 2.17 (Local Convergence of the SQP Method)
Let (x^*,λ^*,μ^*) be a KKT-point of problem (NLP) which satisfies the strict sufficient optimality conditions in Lemma 2.13. Furthermore the regularity LICQ in Definition 2.9 are fulfilled.
Then there exist $\varepsilon > 0$ and a neighbourhood $S_\varepsilon(x^*,\lambda^*,\mu^*)$ with radius ε around (x^*,λ^*,μ^*), such that for every starting point $\left(x^{[0]},\lambda^{[0]},\mu^{[0]}\right)$ and every sequence $\left\{\left(x^{[k]},\lambda^{[k]},\mu^{[k]}\right)\right\}$ generated by the SQP-method using the condition in Note 2.16, the following holds:

i) The local SQP-method is well defined and the sequence $\left\{\left(x^{[k]}, \lambda^{[k]}, \mu^{[k]}\right)\right\}$ converges towards the KKT-point (x^*, λ^*, μ^*).

ii) The method converges superlinearly.

iii) If the second derivatives $\nabla^2_{xx} f$, $\nabla^2_{xx} g$, $\nabla^2_{xx} h$ are locally Lipschitz-continuous, the method converges quadratically.

Proof: Generally speaking, the idea of the proof is analogous to the derivation of the SQP method at the beginning of this section. Due to the strict complementarity conditions, the inactive constraints can be practically ignored. The SQP iterations of the resulting restricted problem then correspond to the iterations of Newton's method.

Formally, the proof is carried out by defining the function

$$\Phi(x, \lambda, \mu) := \begin{pmatrix} \nabla_x L(x, \lambda, \mu) \\ h(x) \\ \min\{-g(x), \lambda\} \end{pmatrix}, \tag{2.26}$$

which is such that every point (x^*, λ^*, μ^*) solving

$$\Phi(x, \lambda, \mu) = 0,$$

is a KKT-point for Problem (NLP). The properties in Theorem 2.17 then follow from the properties of Newton's method applied to Φ. The main part of the proof consists of formally proving that the sequence generated by the SQP method is identical to the sequence of Newton's method. The details are rather technical. Geiger, and Kanzow [53, Theorem 5.31, p. 245-249] provide a comprehensive description. □

Theorem 2.17 proves the convergence of the SQP method by showing that it is equivalent to a Newton iteration for the function Φ (2.26). However, in its basic form, Newton's method only guarantees convergence when starting from a specific neighbourhood of the minima. Thus, the convergence is only local. This property is inherited by the SQP method when used in the raw form described above.

Since the neighbourhood from which an initial guess would converge to the minima is generally unknown, the SQP method in its raw form would not be applicable for most real-world applications. Thus, several different globalisation strategies were discovered in the past. They all work by restricting the stepsize $\left\|d^{[k]}\right\|$ to a specific value. The challenge is to find a reasonable restriction that guarantees global convergence but does not slow down the algorithm too much. One type of globalisation strategy are so-called trust-region algorithms where the value of a model function is compared to the value of a penalty function leading to a radius in which the next step is "trustworthy". The radius is then given as an additional constraint to the QP solver to restrict $\left\|d^{[k]}\right\|$. The book of Conn, Gould and Toint [23] provides good in-depth descriptions of several trust-region methods.
Another type of globalisation strategies are so-called line-search methods. These methods aim at finding a stepsize $\alpha^{[k]} \in (0, 1)$ which leads to a sufficient descent for $x^{[k+1]}$ when applied as

$$x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}. \tag{2.27}$$

One way to achieve this is to evaluate a performance criterion that captures the objective function and constraints in one single value along $d^{[k]}$. The books by Geiger and Kanzow [53],

Nocdeal and Wright [95], and Alt [2] provide a good overview of several performance criteria and the respective algorithms. Another method is to distinguish between the performance related to the objective function and the performance related to the constraints. In this way, it is possible to accept a stepsize that reduces either the objective function or the constraints. These so-called filter methods were introducted by Fletcher and Leyffer [42]. The local convergence properties were afterwards investigated by Ulbrich [115] as well as Wächter and Biegler [119].
Line-search methods aim at maximising the descent achieved in each iteration. However, to find the exact optimum for each line search, it would be necessary to solve another optimisation problem in each iteration of the SQP method. To prevent this, the Armijo-rule [3] helps by determining the stepsize through solving the problem inexactly.

Note 2.18
This section distinguishes local and global convergence. Global convergence in this context means that the algorithm will converge for an initial guess and is not restricted to initial guesses in a local neighbourhood of the minima. However, it still converges to a local minimum. Thus the term global convergence should not be confused with global optimisation solver, which aims at finding the global minimum.

2.2 Parametric Sensitivity Analysis

Section 2.1 introduces the theory of nonlinear optimisation for a problem of type (NLP). However, these problems often depend on a parameter p, which is considered constant during the optimisation but might change in reality. This change is often due to parameters that are not known precisely during optimisation. Thus, they introduce some uncertainty that results from a randomly distributed value, for instance, the wind speed in a dynamical system for ships, but also due to incomplete information, for example, when considering the mass of a ship which might change due to fuel consumption or differences in freight. They can also be arbitrary hyperparameters for which the influence is to be computed, as is done later in Chapter 3, to interpolate between different solutions.
This section introduces the theory of parametric sensitivity analysis, which makes it possible to determine the influence of these so-called disturbance parameters p onto the optimal solution. The underlying theory was mainly developed by Fiacco and is described in detail in his publications [36–39].

2.2.1 Fundamentals of Parametric Sensitivity Analysis

To define the influence of disturbance parameters onto the optimal solution, they need to be added to the formulation of the optimisation problem.

Definition 2.19 (Perturbed Standard Nonlinear Optimisation Problem)
Let the dimensions $0 < N_x \in \mathbb{N}$, $0 \leq N_g \in \mathbb{N}$, $0 \leq N_h \in \mathbb{N}$ and $0 \leq N_p \in \mathbb{N}$ be given. For the functions $f : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}$, $g : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}^{Ng}$ and $h : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}^{N_h}$, and $x \in \mathbb{R}^{N_x}$, the **perturbed Standard Nonlinear Optimisation Problem** (NLP(p)) at $p \in \mathbb{R}^{N_p}$ is defined by

$$\begin{aligned} \min_{x\in\mathbb{R}^{N_x}} \quad & f(x,p), \\ \text{subject to} \quad & g_i(x,p) \leq 0, \quad i = 1,\dots,N_g, \\ & h_j(x,p) = 0, \quad j = 1,\dots,N_h. \end{aligned} \tag{NLP(p)}$$

For the nominal disturbance $p_0 \in \mathbb{R}^{N_p}$, the problem (NLP(p)) is called undisturbed problem or nominal problem.

A look at Definition 2.19 shows that the optimal solution depends implicitly on p. Thus it is necessary to find a way of computing this implicit influence around the nominal parameter p_0. At the heart of the theory of parametric sensitivity analysis is the implicit function theorem, which provides a tool to investigate the implicit dependencies of the optimal solution.

Theorem 2.20 (Implicit Functions)
Let $F : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}^{N_x},\ (x,p) \mapsto F(x,p)$ be a continuously differentiable function. And let $(x^*, p_0) \in R^{N_x} \times \mathbb{R}^{N_p}$ be a point satisfying $F(x^*, p_0) = 0$ and such that the Jacobi matrix $\nabla_x F(x^*, p_0) \in \mathbb{R}^{N_x \times N_x}$ be invertible. Then there exists an open neighbourhood $P \subset \mathbb{R}^{N_p}$ around p_0, a neighbourhood $S \subset \mathbb{R}^{N_x}$ around x^*, as well as a continuously differentiable function $x : P \to S,\ p \mapsto x(p)$ with $x(p_0) = x^*$ and $F(x(p), p) = 0$ for all $p \in P$.
If $(x,p) \in S \times P$ is a point with $F(x,p) = 0$, then $x = x(p)$ follows. In addition, the following applies to the Jacobi matrix of the implicit function in p_0:

$$\frac{dx}{dp}(p_0) = -(\nabla_x F(x^*, p_0))^{-1} \nabla_p F(x^*, p_0). \tag{2.28}$$

Proof: Forster gives the proof in his book [48]. □

Before applying Theorem 2.20 to the optimal solution of problem (NLP(p)), an important lemma is formulated. This lemma provides the property that the KKT-matrix is invertible, which is important later (see [49] for more details).

Lemma 2.21
Let the point (x^*, λ^*, μ^*) satisfy the strict sufficient optimality condition in Lemma 2.13 and therefore the conditions in Theorem 2.12 for the parameter perturbed problem (NLP(p_0)). Then the matrix

$$\mathcal{M} := \begin{pmatrix} \nabla^2_{xx} L(x^*, \lambda^*, \mu^*, p_0) & \nabla_x g(x^*, p_0)^\top & \nabla_x h(x^*, p_0)^\top \\ \Lambda^* \nabla_x g(x^*, p_0) & \Gamma^* & 0 \\ \nabla_x h(x^*, p_0) & 0 & 0 \end{pmatrix}, \tag{2.29}$$

with

$$\Lambda^* := \begin{pmatrix} \lambda_1^* & & \\ & \ddots & \\ & & \lambda_{N_g}^* \end{pmatrix}, \tag{2.30}$$

and

$$\Gamma^* := \begin{pmatrix} g_1(x^*, p_0) & & \\ & \ddots & \\ & & g_{N_g}(x^*, p_0) \end{pmatrix}, \tag{2.31}$$

is invertible.

Proof: First, without loss of generality sort the inequality constraints g in active and inactive ones such that the active set becomes $\mathcal{A}(x^*) = \{k+1, \ldots, N_g\}$, where $k \geq 0$ is the number of inactive constraints. In this way, due to the strict complementary conditions in Lemma 2.13, the matrices Λ^* and Γ^* become

$$\Lambda^* = \begin{pmatrix} 0 & 0 \\ 0 & \Lambda_2^* \end{pmatrix}, \text{ with } \Lambda_2^* = \begin{pmatrix} \lambda_{k+1}^* & & \\ & \ddots & \\ & & \lambda_{N_g}^* \end{pmatrix} \text{ and } \tag{2.32}$$

$$\Gamma^* = \begin{pmatrix} \Gamma_1^* & 0 \\ 0 & 0 \end{pmatrix}, \text{ with } \Gamma_1^* = \begin{pmatrix} g_1(x^*, p_0) & & \\ & \ddots & \\ & & g_k(x^*, p_0) \end{pmatrix}, \tag{2.33}$$

where Λ_2^* and Γ_1^* are invertible.
The following notation is introduced in order to simplify the subsequent expressions:

$$\mathcal{B} := \nabla_{xx}^2 L(x^*, \lambda^*, \mu^*), \tag{2.34}$$
$$\mathcal{C} := (\nabla_x g_1, \ldots, \nabla_x g_k)(x^*), \tag{2.35}$$
$$\mathcal{D} := (\nabla_x g_{k+1}, \ldots, \nabla_x g_{N_g}, \nabla_x h)(x^*). \tag{2.36}$$

By dividing the rows in $\mathcal{M}$ of (2.29) with the corresponding multipliers the matrix becomes

$$\begin{pmatrix} I_{N_x+k} & 0 & 0 \\ 0 & (\Lambda_2^*)^{-1} & 0 \\ 0 & 0 & I_{N_h} \end{pmatrix} \mathcal{M} = \begin{pmatrix} \mathcal{B} & \mathcal{C} & \mathcal{D} \\ 0 & \Gamma_1^* & 0 \\ \mathcal{D}^\top & 0 & 0 \end{pmatrix}. \tag{2.37}$$

Let $v \in \mathbb{R}^{N_x+N_g+N_h}$ be appropriately partitioned with $v_1 \in \mathbb{R}^{N_x}$, $v_2 \in \mathbb{R}^k$ and $v_3 \in \mathbb{R}^{(N_g-k)+N_h}$. To show that $\mathcal{M}$ is invertible, it suffices to verify that its kernel is trivial, that is, $\mathcal{M}v = 0$ implies that $v = 0$. With Equation (2.37), it follows from $\mathcal{M}v = 0$ that the relations

$$\mathcal{B}v_1 + \mathcal{C}v_2 + \mathcal{D}v_3 = 0, \tag{2.38}$$
$$\Gamma_1^* v_2 = 0, \tag{2.39}$$
$$\mathcal{D}^\top v_1 = 0, \tag{2.40}$$

hold. Since the strict complementary conditions (2.15) are fulfilled, it follows directly from Equation (2.39) that $v_2 = 0$. Applied to (2.38) this results in

$$\mathcal{B}v_1 + \mathcal{D}v_3 = 0. \tag{2.41}$$

Multiplying this last equation by $v_1^\top$ and combining with (2.40) yields

$$0 = v_1^\top \mathcal{B}v_1 + v_1^\top \mathcal{D}v_3 = v_1^\top \mathcal{B}v_1 + \left(\mathcal{D}^\top v_1\right)^\top v_3 = v^\top \mathcal{B}v_1, \tag{2.42}$$

from which, on the basis of positive definiteness of $\mathcal{B}$, it follows that $v_1 = 0$. Thus, (2.38) becomes

$$\mathcal{D}v_3 = 0, \tag{2.43}$$

and since $\mathcal{D}$ is invertible, it follows that $v_3 = 0$. In total, this results in $v = 0$ from which the invertibility of $\mathcal{M}$ follows. □

With this result the central result of this section is formulated (see [18, 38, 49]).

Theorem 2.22 (Sensitivity Theorem)
Let $f(x,p) : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}$, $g(x,p) : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}^{N_g}$ and $h(x,p) : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \to \mathbb{R}^{N_h}$ be twice continuously differentiable with respect to $x \in \mathbb{R}^{N_x}$. In addition, the functions g and h, as well as $\nabla_x f$, $\nabla_x g$ and $\nabla_x h$ be continuously differentiable with respect to p.
Let (x^*, λ^*, μ^*) with $x^* \in \Sigma(p_0)$ be a point which satisfies the strict sufficient optimality conditions in Lemma 2.13 for problem NLP(p_0).
Then there exist a neighbourhood $P \in \mathbb{R}^{N_p}$ around p_0 and continuously differentiable functions $x : P \to \mathbb{R}^{N_x}$, $\lambda : P \to \mathbb{R}^{N_g}$ and $\mu : P \to \mathbb{R}^{N_h}$ to which the following statements apply:

i) $x(p_0) = x^*$, $\lambda(p_0) = \lambda^*$, $\mu(p_0) = \mu^*$.

ii) The set of active indices remains unchanged:

$$\mathcal{A}(x(p)) = \mathcal{A}(x(p_0)) \text{ for all } p \in P.$$

iii) The point $(x(p), \lambda(p), \mu(p))$ satisfies the strict sufficient optimality conditions from Lemma 2.13 for all $p \in P$ and, in particular, remains a strict local minimum of NLP(p).

Proof: First define the auxiliary function $F(\zeta, p)$ with $\zeta := (x, \lambda, \mu)$ and rewrite the optimality condition (2.5), the complementary condition (2.8) and the equality constraints as

$$F(\zeta, p) := \begin{pmatrix} \nabla_x L(x, \lambda, \mu, p) \\ \Lambda g(x,p) \\ h(x,p) \end{pmatrix} = 0. \tag{2.44}$$

Since the strict sufficient optimality conditions are fulfilled for (x^*, λ^*, μ^*) the statement is satisfied for $p = p_0$ and $\zeta^* = (x^*, \lambda^*, \mu^*)$. Under the given conditions, Lemma 2.21 is applicable and thus, the matrix

$$\nabla_\zeta F(\zeta^*, p_0) = \mathcal{M} = \begin{pmatrix} \nabla^2_{xx} L(x^*, \lambda^*, \mu^*, p_0) & \nabla_x g(x^*, p_0)^\top & \nabla_x h(x^*, p_0)^\top \\ \Lambda^* \nabla_x g(x^*, p_0) & \Gamma^* & 0 \\ \nabla_x h(x^*, p_0) & 0 & 0 \end{pmatrix}, \tag{2.45}$$

is invertible. Therefore all prerequisites for Theorem 2.20 are fulfilled and for all $p \in P$, there exists exactly one $\zeta(p)$ such that $F(\zeta(p), p) = 0$ and therefore, the KKT Conditions of (NLP(p)) are satisfied. In addition, it follows from Theorem 2.20 that $\zeta(p)$ is differentiable with respect to p and it can be stated that

$$\frac{d\zeta}{dp}(p_0) = \begin{pmatrix} \frac{dx}{dp}(p_0) \\ \frac{d\lambda}{dp}(p_0) \\ \frac{d\mu}{dp}(p_0) \end{pmatrix} = -\nabla_\zeta F(\zeta^*, p_0)^{-1} \nabla_p F(\zeta^*, p_0). \tag{2.46}$$

Finally, it is shown that $x(p)$ satisfies the strict complementary conditions (2.15) and that the constraints qualification from Definition 2.9 still holds. Without loss of generality the constraint functions can be sorted as in the proof of Lemma 2.21. Thus, the inequalities $\lambda^*_{k+1}, \dots, \lambda^*_{N_g} > 0$ and $g_1(x^*, p_0), \dots, g_k(x^*, p_0) < 0$ hold. Since g and λ are continuous with respect to p it follows that $\lambda_{k+1}(p), \dots, \lambda_{N_g}(p) > 0$ and $g_1(x(p), p), \dots, g_k(x(p), p) < 0$ for $p \in P$, where P is a sufficiently small neighbourhood of p_0. In addition,

$$g_{k+1}(x(p), p) = \ldots = g_{N_g}(x(p), p) = 0, \tag{2.47}$$

and

$$\lambda_1(p) = \dots \lambda_k(p) = 0, \tag{2.48}$$

apply as $F(\zeta(p), p) = 0$ applies. This results in the fulfilment of strict complementarity conditions (2.15). It also follows directly that the active set remains unchanged:

$$\mathcal{A}(x(p_0)) = \mathcal{A}(x(p)). \tag{2.49}$$

The evidence that the LICQ 2.9 also holds similarly follows from the continuity of the functions and derivatives regarding p involved. □

Theorem 2.22 provides an important tool to gather more insight into the problem of type (NLP(p)). It shows the general dependency of the optimal solution on the disturbance parameter. An important tool for the techniques described in this thesis are sensitivity differentials that are given through (2.46) but should be recalled at this point:

$$\begin{pmatrix} \frac{dx}{dp}(p_0) \\ \frac{d\lambda}{dp}(p_0) \\ \frac{d\mu}{dp}(p_0) \end{pmatrix} = -\begin{pmatrix} \nabla^2_{xx}L(\zeta^*, p_0) & \nabla_x g(x^*, p_0)^\top & \nabla_x h(x^*, p_0)^\top \\ \Lambda^* \nabla_x g(x^*, p_0) & \Gamma^* & 0 \\ \nabla_x h(x^*, p_0) & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla^2_{xp}L(\zeta^*, p_0) \\ \Lambda^* \nabla_p g(x^*, p_0) \\ \nabla_p h(x^*, p_0) \end{pmatrix}. \tag{2.50}$$

Based on these differential, Taylor's Theorem is used to formulate linear approximations of the primal and dual variables in the optimisation problem:

$$x(p) \approx x(p_0) + \frac{dx}{dp}(p_0)\,(p - p_0)\,, \tag{2.51}$$

$$\lambda(p) \approx \lambda(p_0) + \frac{d\lambda}{dp}(p_0)\,(p - p_0)\,, \tag{2.52}$$

$$\mu(p) \approx \mu(p_0) + \frac{d\mu}{dp}(p_0)\,(p - p_0)\,. \tag{2.53}$$

With these approximations, it is possible to compute updated optimal solutions for a small change in the parameter. With this new solution, the new value for the objective function can be computed. However, it is also of interest to get the sensitivity differential for the objective function directly for the parameter. To get these, an affiliation function is defined which was introduced by Büskens [16].

Definition 2.23
Let $\tilde{k} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_g} \times \mathbb{R}^{N_h} \times \mathbb{R}^{N_p} \to \mathbb{R}$ be a twice continuously differentiable function. In addition there is a nominal perturbation $p_0 \in \mathbb{R}^{N_p}$ given and the conditions of the sensitivity theorem 2.22 are fulfilled in a neighbourhood $P \subset \mathbb{R}^{N_p}$ of p_0 for the continuously differentiable functions $x : P \to \mathbb{R}^{N_x}$, $\lambda : P \to \mathbb{R}^{N_g}$ and $\mu : P \to \mathbb{R}^{N_g}$. Then $k(p) := \tilde{k}(x(p), \lambda(p), \mu(p), p)$ is called an **affiliation function** of the problem (NLP(p)).

Note 2.24
Let the functions f, g and h be continuously differentiable with respect to p. Then they are also affiliation functions of the problem (NLP(p)). Furthermore, the Lagrange function in this case is also an affiliation function.

By using these affiliation functions and applying the Sensitivity Theorem to them, the influence of the disturbances on them are computed. The arguments of the functions are omitted in the following if they are obvious from the context to make the content more readable.

Lemma 2.25
Let an affiliation function $k : P \to \mathbb{R}$ be given and the conditions of Theorem 2.22 be fulfilled. Then the sensitivity differential of the affiliation function can be determined by

$$\frac{dk}{dp}(p_0) = (\nabla_x k(p_0), \nabla_\lambda k(p_0), \nabla_\mu k(p_0)) \begin{pmatrix} \frac{dx}{dp}(p_0) \\ \frac{d\lambda}{dp}(p_0) \\ \frac{d\mu}{dp}(p_0) \end{pmatrix} + \nabla_p k(p_0). \tag{2.54}$$

Proof: The statement follows directly from application of the chain rule and Theorem 2.20. □

With this result, the sensitivities for the constraints can be derived as well as the sensitivities of the Lagrange function. Based on that, the sensitivities of the objective function can be furthermore derived (see [16]).

Corollary 2.26 (Sensitivities of constraint functions)
Let the conditions of Theorem 2.22 be satisfied, then

$$\frac{dh_j}{dp}(p_0) = \nabla_x h_j(x^*, p_0)\frac{dx}{dp}(p_0) + \nabla_p h_j(x^*, p_0) = 0, \quad j = 1, \dots, N_h, \tag{2.55}$$

$$\frac{dg_i}{dp}(p_0) = 0, \quad i \in \mathcal{A}(x(p_0)), \tag{2.56}$$

$$\frac{dg_i}{dp}(p_0) = \nabla_x g_i(x^*, p_0)\frac{dx}{dp}(p_0) + \nabla_p g_i(x^*, p_0), \quad i = 1, \dots, N_g,\ i \notin \mathcal{A}(x(p_0)), \tag{2.57}$$

$$\frac{d\lambda_i}{dp}(p_0) = 0, \quad i = 1, \dots, N_g,\ i \notin \mathcal{A}(x(p_0)), \tag{2.58}$$

apply.

Proof: The proof foolows from Lemma 2.25 and the fact that the active set does not change for all $p \in P$. □

Corollary 2.27 (Sensitivities of the Lagrange function)
Let the conditions of Theorem 2.22 be satisfied. In addition let f be continuously differentiable with respect to p. Then for all $p \in P$ the equation

$$\frac{dL}{dp}(x^*, \lambda^*, \mu^*, p_0) = \nabla_p L(x^*, \lambda^*, \mu^*, p_0), \tag{2.59}$$

applies.

Proof: The Lagrange function is an affiliation function in the sense of Definition 2.23. In addition, the equations $\nabla_x L(x^*, \lambda^*, \mu^*, p_0) = 0$, $\nabla_\lambda L(x^*, \lambda^*, \mu^*, p_0) = g(x^*, p_0)$, and $\nabla_\mu L(x^*, \lambda^*, \mu^*, p_0) = h(x^*, p_0) = 0$ apply. Then, by applying Lemma 2.25, it follows that

$$\begin{aligned}
\frac{dL}{dp}(x^*, \lambda^*, \mu^*, p_0) &= (\nabla_x L, \nabla_\lambda L, \nabla_\mu L)(x^*, \lambda^*, \mu^*, p_0) \begin{pmatrix} \frac{dx}{dp}(p_0) \\ \frac{d\lambda}{dp}(p_0) \\ \frac{d\mu}{dp}(p_0) \end{pmatrix} + \nabla_p L(x^*, \lambda^*, \mu^*, p_0) \\
&= \nabla_\lambda L(x^*, \lambda^*, \mu^*, p_0)\frac{d\lambda}{dp}(p_0) + \nabla_p L(x^*, \lambda^*, \mu^*, p_0) \\
&= g(x^*, p_0)\frac{d\lambda}{dp}(p_0) + \nabla_p L(x^*, \lambda^*, \mu^*, p_0) \\
&= \nabla_p L(x^*, \lambda^*, \mu^*, p_0),
\end{aligned}$$

whereby the last equation holds because of the strict complementarity condition in Lemma 2.13 combined with Equation (2.58) which yield $g_i(x^*, p_0) = 0$ for $i \in \mathcal{A}(x^*)$ and $\frac{d\lambda_i}{dp}(p_0) = 0$ for $i \notin \mathcal{A}(x^*)$. □

Corollary 2.28 (First Order Sensitivity of the Objective Function)
Let the conditions of Theorem 2.22 be satisfied. In addition let f be continuously differentiable with respect to p. Since the objective function is an affiliation function of problem (NLP(p_0)), it follows from Equation (2.54) that

$$\frac{df}{dp}(x^*, p_0) = (\nabla_x f(x^*, p_0), 0, 0) \begin{pmatrix} \frac{dx}{dp}(p_0) \\ \frac{d\lambda}{dp}(p_0) \\ \frac{d\mu}{dp}(p_0) \end{pmatrix} + \nabla_p f(x^*, p_0) \tag{2.60}$$

$$= (\lambda^*)^\top \nabla_p g(x^*, p_0) + (\mu^*)^\top \nabla_p h(x^*, p_0) + \nabla_p f(x^*, p_0) = \nabla_p L(x^*, \lambda^*, \mu^*, p_0). \tag{2.61}$$

Proof: The first equation follows directly from (2.54). For the deduction of the second relationship, it should be recalled that $\nabla_x f(x^*, p_0) = -(\lambda^*)^\top \nabla_p g(x^*, p_0) - (\mu^*)^\top \nabla_p h(x^*, p_0)$ follows from the KKT-condition (2.5). In addition, it follows from the strict complementarity conditions that $\lambda_i^* = 0$, $\forall i \notin \mathcal{A}(x^*)$. For $i \in \mathcal{A}(x^*)$ it follows from (2.57) and (2.56) that $-\nabla_x g(x^*, p_0)\frac{dx}{dp}(p_0) = \nabla_p g(x^*, p_0)$ and from (2.55) that $-\nabla_x h(x^*, p_0)\frac{dx}{dp}(p_0) = \nabla_p h(x^*, p_0)$. By using these relations in (2.60) as well as (2.61) the proposition results. □

Note 2.29
Equations (2.60) and (2.61) show the relation between the local behaviour of the objective function and the behaviour of the Lagrangian with respect to the parameter p. Büskens [17,18] also notes that the equations are independent of the inverted KKT-Matrix, which makes the value of $\frac{df}{dp}(x^*, p_0)$ as well as $\frac{dL}{dp}(x^*, \lambda^*, \mu^*, p_0)$ cheap in computation if $\nabla_p f(x^*, p_0)$ and $\nabla_p g(x^*, p_0)$ are cheap in computation as well.

2.2.2 Additive Perturbations in the Constraint Functions

The equations in Section 2.2.1 show the correlations for general perturbations p. However, they can be simplified for several perturbations that have a special structure. One of special structures is given in the case where $p = q \in \mathbb{R}^{N_g+N_h}$ is simply added to the constraints such that (NLP(p)) becomes

$$\begin{array}{lll} \min\limits_{x\in\mathbb{R}^{N_x}} & f(x), & \\ \text{subject to} & g_i(x) + q_i \le 0, & i = 1, \ldots, N_g, \\ & h_j(x) + q_{N_g+j} = 0, & j = 1, \ldots, N_h. \end{array} \tag{2.62}$$

With $\tilde{g}_i(x, q) = g_i(x) + q_i$ and $\tilde{h}_j(x, q) = h_j(x) + q_{N_g+j}$ it follows that $\nabla_q \tilde{g}(x^*, q_0) = (1, \ldots, 1)^\top$ as well as $\nabla_q \tilde{h}(x^*, q_0) = (1, \ldots, 1)^\top$. Together with equations (2.60) and (2.61), this implies that

$$\frac{df}{dq}(x^*, q_0) = \frac{dL}{dq}(x^*, \lambda^*, \mu^*, q_0) = (\lambda_1^*, \ldots, \lambda_{N_g}^*, \mu_1^*, \ldots, \mu_{N_h}^*)^\top. \tag{2.63}$$

Furthermore the right-hand side of Equation (2.50) becomes

$$\begin{pmatrix} \nabla_{xp}^2 L(x^*, \lambda^*, \mu^*, q_0) \\ \Lambda^* \nabla_q \tilde{g}(x^*, q_0) \\ \nabla_q \tilde{h}(x^*, q_0) \end{pmatrix} = \begin{pmatrix} 0 \\ \Lambda^* \\ I_{N_h} \end{pmatrix}. \tag{2.64}$$

with the N_h-dimensional unity matrix $I_{N_h} \in \mathbb{R}^{N_h \times N_h}$. Therefore, Equation (2.50) can be calculated without additional computational effort [17, 18]. Büskens states that perturbations which occur as additive perturbations as in (2.62) can be interpreted as a violation of feasibility. This can be exploited in a real-time feasibility correction algorithm, see [17, 76, 89, 94]. The perturbations are also important in the development of the adaptive scalarisation algorithm for multi-objective problems (see [33]) and for the Pareto front interpolation in Chapter 3.

2.3 The NLP-Solver WORHP

Section 2.1.2 explains the theoretical idea and fundamentals of Sequential Quadratic Programming (SQP). This section introduces the NLP solver WORHP ("We Optimise Really Huge Problems") [19] which implements the ideas for the SQP method to solve optimisation problems numerically. However, there are additional details and differences in the formulation, which are explained in the following sections.

Originally WORHP was implemented as an SQP based solver [19, 120] for space applications and chosen by ESA as the European NLP solver based on its high robustness and its application-driven design. Nowadays it is used in a wide range of applications, such as research projects ranging from autonomous driving [106, 111], autonomous ship manoeuvring [52], space-trajectory optimisation [102], electric grid optimisation [6], energy system controlling [21, 69, 85] and parameter identification [77] as well as industrial projects and commercial use.

Note 2.30
For some years now, there has also been an implementation of a Penalty-Interior-Point algorithm for WORHP [78]. However, the work in this thesis was done using the SQP algorithm. Thus, this thesis provides no further details on this algorithm. For the interested reader, Kuhlmann [78] gives an overview of the algorithm implemented in WORHP.

The problem formulation for an NLP within WORHP differs a bit from the one in (NLP). More specifically WORHP distinguishes between constraints directly given for the variables, so-called box-constraints and the usual nonlinear constraints $g(x)$. The box-constraints are of the type $x_l \geq x_l^{\text{lower}}$ and $x_l \leq x_l^{\text{upper}}$ for $l = 1, \ldots, N_x$ and given numbers $x_l^{\text{lower}} \in \mathbb{R}$ and $x_l^{\text{upper}} \in \mathbb{R}$. Also, the formulation WORHP uses does not distinguish between equality and inequality constraints but defines a lower bound $g_i^{\text{lower}} \in \mathbb{R}$ and an upper bound $g_i^{\text{upper}} \in \mathbb{R}$ for each constraint $i = 1, \ldots, N_g$. Equality constraints are then characterised by $g_i^{\text{lower}} = g_i^{\text{upper}}$ whereas the fact that a constraint, either box-constraint or nonlinear, has no lower or upper limit is represented by choosing the respective value as infinity. Within the code implementation this is done by setting the lower or upper bound to be higher / lower as a specific WORHP hyperparameter which represents infinity. This parameter is by default set to 10^{20}. Altogether, this leads to the formulation

$$\begin{array}{lll} \min\limits_{x \in \mathbb{R}^{N_x}} & f(x,p), & \\ \text{subject to} & x_l \leq x_l^{\text{upper}}, & l = 1, \ldots, N_x, \\ & x_l \geq x_l^{\text{lower}}, & l = 1, \ldots, N_x, \\ & g_i(x,p) \leq g_i^{\text{upper}}, & i = 1, \ldots, N_g, \\ & g_i(x,p) \geq g_i^{\text{lower}}, & i = 1, \ldots, N_g, \end{array} \qquad \text{(WORHP NLP)}$$

where $p \in \mathbb{R}^{N_p}$ is the perturbation which is introduced in Section 2.2.1 and is set to the nominal parameter p_0 for the optimisation itself.

To achieve global convergence, WORHP implements line search methods as described in Section 2.1.2. The original techniques used are described by Wassel [120]. They implement the line-search method using a performance criterion for the objective function and constraints simultaneously. Later Kemper [72] adapted and implemented the filter methods for WORHP based on the work of Fletcher and Leyffer [42].

Explicitly designed for high-dimensional optimisation problems, WORHP is characterised by exploiting sparse structures within the derivative matrices, notably, the gradient of the objective function, but more importantly, the Jacobi matrix of the constraints and the Hesse matrix of the Lagrange function [120]. This sparsity leads to a memory-efficient implementation and also to numerically more efficient algorithms for solving the internal QP and linear equation systems during the optimisation.

To compute derivatives of the objective function and constraints, WORHP provides an interface for the user to compute the derivatives analytically or let WORHP compute them via finite differences. To reduce function calls and thus improve the efficiency of the computation via finite differences, WORHP uses methods from graph-theory [70]. In addition, the computation of the Hesse matrix of the Lagrangian can be done analytically, via finite-differences or by using the Broyden-Fletcher-Goldfarb-Shanno-method (BFGS) for an approximation [56,70,104], which was developed independently from each other by Broyden [15], Fletcher [40], Goldfarb [57], and Shanno [110].

Note 2.31
The BFGS-method is often a good way to accelerate the optimisation process, especially when no analytical derivative of the Hesse matrix of the Lagrangian is given since it does need significantly fewer function evaluations than finite-differences. However, when using parametric sensitivities afterwards, the exact Hesse matrix must be computed analytically or via finite differences.

Further developments in the implementation of WORHP were made by Geffken [49], Geffken and Büskens [51] as well as Jacobse and Büskens [67]. Geffken and Büskens [50] also present a way of utilising parallelisation for the optimisation process by starting several instances of WORHP in parallel and using the best or fastest solution from all runs. However, this advancement is not part of the solver WORHP itself but is implemented as external code. Throughout this thesis, several attempts are made to parallelise the computation of Pareto fronts by calling WORHP in parallel. This parallelisation is motivated by the work in [50].

To compute parametric sensitivities efficiently, for example by exploiting the factorisation of the KKT-matrix, which is already done during the iteration process, WORHP offers the module WORHP Zen [80], which is directly integrated into the solver.

Another external library in the ecosystem of WORHP is TransWORHP a transcription method presented by Knauer and Büskens in [74]. TransWORHP offers an interface to discretise and solve optimal control problems via direct methods.
There exists also a program called WORHPLab [75] which is an easy-to-use tool that helps by discovering the possibilities of optimisation and optimal control.
The recently developed tool TOPAS-Model-Fitting [124] is also based on WORHP. It is a tool that utilises the features of WORHP and WORHP Zen [125] to improve the process of parameter identification of dynamical systems.

Altogether, WORHP implements an efficient and robust NLP solver that offers a variety of possibilities to the user to adjust the optimisation process based on the specific needs of a problem. This thesis extends the basic WORHP implementation by incorporating another external library that offers an interface to treat multi-objective optimisation problems. How this is done is described throughout the next section and chapters. Chapter 6.1 also explains the architecture and some implementation details for the interface and how a library like TransWORHP can be integrated to be able to solve multi-objective optimal control problems. The code for this is available in [4].

2.4 Fundamentals of Nonlinear Multi-Objective Optimisation

So far, the last section has characterised optimal solutions by being better than all other points around them concerning a specific objective function. However, often and especially in real-world applications, it is impossible to model the goal of the decision-maker in one single-objective function. If, for example, a ship is controlled towards a particular target, energy consumption is one objective to optimise. However, if the optimisation considers energy consumption alone, the optimal solution might be that the ship does not start to move. Of course, appropriate constraints can prevent this. Nevertheless, in reality, the bounds of the constraints might not be known, which may lead to sub-optimal solutions. In addition, and more importantly, the addition of constraints is already one method of dealing with multiple objectives, known as the ε-constraint method. In fact, all constrained optimisation problems somehow represent a multi-objective optimisation problem.
Another widespread way of dealing with multiple objectives is to optimise the sum of all objectives. This so-called weighted sum approach is another well-known method and is also part of the theory of multi-objective optimisation.
These examples show that many optimisation processes already involve multi-objective optimisation, often without an explicit treatment of the challenges arising. However, these challenges might lead to unexpected problems when the theory of multi-objective optimisation is not considered appropriately.

This section introduces the fundamentals of multi-objective optimisation. First, the concept of optimality is adapted to the situation where more than one objective is considered. The theoretical fundamentals are covered briefly, and the main concepts are explained. Methods to solve multi-objective optimisation problems numerically are presented in the next section.

At the heart of multi-objective optimisation, as for optimisation in general, is the concept of optimality. If more than one objective is considered simultaneously, it is not always possible to say if one point is better or worse than another. Mathematically, this can be expressed in the fact that the objective function does not map to $\mathbb{R}$ anymore but instead becomes a function $F : \mathbb{R}^{N_x} \to \mathbb{R}^{N_F}$ with $1 < N_F \in \mathbb{N}$. This formulation leads to the standard problem of multi-objective optimisation.

Definition 2.32 (Standard nonlinear multi-objective optimisation problem)
Let the dimensions $1 < N_F \in \mathbb{N}$, $0 < N_x \in \mathbb{N}$, $0 \leq N_g \in \mathbb{N}$ and $0 \leq N_h \in \mathbb{N}$ be given. For the functions $F : \mathbb{R}^{N_x} \to \mathbb{R}^{N_F}$, $g : \mathbb{R}^{N_x} \to \mathbb{R}^{Ng}$ and $h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_h}$, and $x \in \mathbb{R}^{N_x}$, the **standard**

nonlinear multi-objective optimisation problem (MONLP) is defined as

$$\begin{array}{llcl} \min\limits_{x\in\mathbb{R}^{N_x}} & F(x) & = & (F_1(x), F_2(x), \dots, F_{N_F}(x))^T, \\ \text{subject to} & g_i(x) & \leq & 0, \quad i = 1, ..., N_g, \\ & h_j(x) & = & 0, \quad j = 1, ..., N_h. \end{array} \tag{MONLP}$$

In this context, the function $F(x) = (F_1(x), \dots, F_{N_F}(x))^\top$ is called **objective function** and $F_l(x)$ are the **individual objective functions** for $l = 1, \dots, N_F$. As in (NLP), $g(x)$ describes the **inequality constraints** and $h(x)$ the **equality constraints**.

Together with the fact that there exists no total order for $\mathbb{R}^{N_F}$ if $N_F > 1$, it becomes clear that the notion of optimality as defined in Definition 2.3 cannot be applied to multi-objective optimisation problems.
Several approaches exist to overcome this problem. One is to prioritise the individual objectives, then optimise for the individual objective with the highest priority. Afterwards, the second objective is optimised without worsening the first. This so-called lexicographic optimisation is described by Rentmeesters et al. in [105].
In contrast to lexicographic optimisation, the idea of Pareto-optimality exists. It is credited to the economist Vilfredo Federico Pareto and is central to many multi-objective optimisation processes today. This thesis uses methods based on Pareto-optimality, and this section explains the fundamentals behind this theory. Miettinen gives an overview and provides more details on this topic in [91]. Also, the formulations in this section are mainly based on the book by Miettinen. More information is also given in the books by Ehrgott [31], Chankong and Haimes [20] and Jahn [68].

The basic idea of Pareto-optimality is to define points as optimal for which no other points exist, which have a better value for all objectives. Thus, if switching from one Pareto-optimal point to another, always some tradeoff has to be considered. Usually, there exists a whole set of points fulfilling these criteria. The set is called the Pareto set and is sometimes also called the set of "optimal compromises". However, the word compromise should be carefully considered because a (strict) optimal point for one individual objective is also Pareto-optimal but would not be considered a compromise. In nonlinear continuous optimisation, the Pareto set is most often a set with an infinite number of points. In [91], Miettinen gives a formal definition of the concept of Pareto-optimality.

Definition 2.33 (Pareto-Optimality)
For (MONLP) a point $x^* \in \Sigma$ is called

-Globally Pareto-optimal if and only if there exists no $x \in \Sigma$ with $F_l(x) \leq F_l(x^*)$ for all $l = 1, \dots, N_F$ and $F_m(x) < F_m(x^*)$ for at least one index $m \in \{1, \dots, N_F\}$.

-Locally Pareto-optimal if and only if there exists a neighbourhood $S_{x^*} \subset \mathbb{R}^{N_x}$ around x^* and x^* is globally Pareto-optimal in $\Sigma \cap S_{x^*}$.

Coupled to the idea of Pareto-optimality is the notion of a point dominating another point.

Definition 2.34 (Dominance)
For (MONLP) a point $x \in \Sigma$ dominates another point $y \in \Sigma$ if $F_l(x) \leq F_l(y)$ for all $l = 1, \dots, N_F$ and $F_m(x) < F_m(y)$ for at least one $m \in \{1, \dots, N_F\}$.

Pareto-optimality can also be expressed such that a point is Pareto-optimal if it is not dominated by any other feasible point. Beside the normal Pareto-optimality there exists also the idea of weak Pareto-optimality which is also defined in [91].

Definition 2.35 (Weak Pareto-Optimality)
For (MONLP) a point $x^* \in \Sigma$ is called **weakly Pareto-optimal** if there does not exist another point $x \in \Sigma$ such that $F_l(x) < F_l(x^*)$ for all $l = 1, \ldots, N_F$. If for a neighbourhood $S_{x^*} \subset \mathbb{R}^{N_x}$ around x^* the vector x^* is weakly Pareto-optimal in $\Sigma \cap S_{x^*}$ it is called **locally weakly Pareto-optimal**.

In contrast to Pareto-optimality, weakly Pareto-optimal points can be improved by switching to another point without worsening any objective. For a Pareto-optimal point it would always introduce a real tradeoff when switching between different solutions.

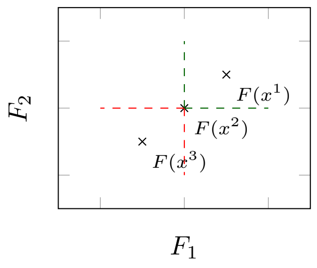


Figure 2.1: Example for Pareto-optimality of point $x^{[3]}$ with respect to F.

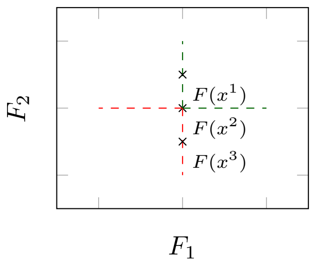


Figure 2.2: Example for weak Pareto-optimality of point $x^{[1]}$ and $x^{[2]}$ with respect to F.

Figures 2.1 and 2.2 visualise the idea of Pareto-optimality and weak Pareto-optimality for the bi-objective case with $N_F = 2$. In Figure 2.1 the point $x^{[2]}$ dominates $x^{[1]}$ and is dominated by $x^{[3]}$. In this scenario only $x^{[3]}$ is Pareto-optimal. In contrast, in Figure 2.2 $x^{[3]}$ is still Pareto-optimal but $x^{[1]}$ and $x^{[2]}$ are also weakly Pareto-optimal. Usually weak Pareto-optimal solutions are not considered as good solutions, since they can be improved in one or more objectives without a tradeoff in another objective. However, some numerical solving methods only guarantee that they find weak Pareto-optimal points.
Figures 2.1 and 2.2 also show another idea behind Pareto-optimality. That is, that the definition of dominance can also be given through cones. A formal definition can for example be found in [68]. The idea is that a point $x \in \mathbb{R}^{N_x}$ dominates another point $y \in \mathbb{R}^{N_x}$ if $F(y)$ is in the set $F(x) + \mathbb{R}_+^{N_F}$ where $\mathbb{R}_+^{N_F}$ is the cone given by $\{z \in \mathbb{R}^{N_F} : z_i \geq 0, \text{ for all } i = 1, \ldots, N_F\}$ and $F(x) + \mathbb{R}_+^{N_F}$ "moves" the cone to the point $F(x)$. The green lines in Figure 2.1 represent this cone. After defining the usual Pareto-optimality through the special cone $\mathbb{R}_+^{N_F}$ it becomes clear that the definition can be given more generally by also allowing other cones. This can lead to advantageous properties of the points which are optimal for different cones [63]. However, this thesis does not exploit these additional possibilities. Thus, to keep equations simple Definition 2.33 is used throughout the following chapters. Details on cone-optimality are given in [11, 62, 64, 68].

As for single-objective optimisation, there exist optimality criteria for points with respect to multiple objectives. In fact, they are analogous to those presented in Section 2.1.1. The basis is formed by the Lagrange function for multi-objective optimisation problems which differs only from the Lagrange function 2.6 only with respect to the additional scalar product for the objective function with a given weight vector w.

Definition 2.36 (Lagrange function for Multi-Objective Optimisation Problems)
For the vectors $w \in \mathbb{R}^{N_F}$, $\lambda \in \mathbb{R}^{N_g}$ and $\mu \in \mathbb{R}^{N_h}$ the function

$$L^{[\text{mo}]}(x, w, \lambda, \mu) := w^\top F(x) + \lambda^\top g(x) + \mu^\top h(x), \tag{2.65}$$

is defined as the **Lagrange function for multi-objective optimisation**. Equivalent to the single-objective Lagrange function (2.4), λ and μ are called the Lagrange multipliers. The vector w is called the weight vector.

Interestingly necessary first-order conditions do not only exist equivalent to the single-objective KKT conditions 2.7 but were also described in the same paper by Kuhn and Tucker [82] and also by Karush (see [71, 81]).

Definition 2.37 (Multi-objective KKT conditions)
A point $(x^*, w^*, \lambda^*, \mu^*) \in \mathbb{R}^{N_x} \times \mathbb{R}^{N_F} \times \mathbb{R}^{N_g} \times \mathbb{R}^{N_h}$ with $\sum_{l=1}^{N_F} w_l = 1$ satisfies the Karush-Kuhn-Tucker conditions if and only if

$$\begin{aligned}
\nabla_x L^{[\text{mo}]}(x^*, w^*, \lambda^*, \mu^*) &= 0, && (2.66)\\
g_j(x^*) &\leq 0, \quad i = 1, \dots, N_g, && (2.67)\\
h_i(x^*) &= 0, \quad j = 1, \dots, N_h, && (2.68)\\
\lambda_i^* g_i(x^*) &= 0, \quad i = 1, \dots, N_g, && (2.69)\\
\lambda_i^* &\geq 0, \quad i = 1, \dots, N_g. && (2.70)
\end{aligned}$$

In this case the point $(x^*, w^*, \lambda^*, \mu^*)$ is called a **KKT-Point**.

Note, that the definition for multi-objective KKT conditions only differs in (2.66) from the definition of the single-objective KKT conditions 2.7 by using the multi-objective Lagrange function 2.36. The remaining equations are the same as for single-objective optimisation. Similarly, the remaining results from Section 2.1.1 are adapted to multi-objective optimisation and are not repeated at this point. For details see the book of Miettinen [91].

Next, it is the task to compute the Pareto set. Since this set is most often a partly continuous set with an infinite amount of points it is usually approximated through sampling. The next section introduces some of the common methods for this task.

2.5 Scalarisation Methods for Multi-Objective Optimisation Problems

Section 2.4 shows that all in all, the optimality conditions for multi-objective optimisation are very similar to those of single-objective optimisation. The process of solving a multi-objective optimisation problem, however, is different from solving single-objective optimisation problems. This difference is directly evident from the fact that for multi-objective optimisation, a whole set with an infinite number of points is usually the solution for a given problem of type (MONLP). This Section focuses on methods that aim to present the decision-maker an approximation of the whole solution set to guide the final decision through scalarisation of the problem (MONLP). However, there exist other approaches to the solution of multi-objective optimisation problems. They are, for example, described in Fliege and Svaiter [45] who describe a descent method based on gradient information, or in Schäffler, Schultz and Weinzierl [108] for stochastic methods.

There exist also a variety of heuristic approaches like the widespread NSGA-II [27]. In [44] Fliege, Grana Drummond and Svaiter develop a Newton-like method that uses scalarised problems for intermediate steps but does not require an up-front scalarisation. This approach is extended in [46] by Fliege and Vaz to deal with general constrained multi-objective optimisation problems. Chapter 4 discusses this approach in more detail and introduces some improvements to the method.

This section describes three different scalarisation approaches. First, the weighted sum approach is presented, which is one of the most common but also often criticised (see [25]). The second approach is the ε-constraint method that is also very common. Both methods are often used in a context where multiple objectives are present, but the decision-maker does not explicitly think of multi-objective optimisation theory. As these approaches are very intuitive, they should always be considered when dealing with multi-objective optimisation problems. For despite some criticism of the approaches, they are often good enough, and the simplicity and intuitiveness make them easy to implement, which is, in real-world problems, often an important property.
The third approach presented in this section is the Pascoletti-Serafini scalarisation. It is more complicated to understand but is regarded as a more sophisticated for complicated problems [32]. It can be considered a generalisation of several approaches [32], including the ε-constraint method, which makes it a good candidate for a general-purpose solver since it can be configured to represent other approaches like the normal-boundary-intersection method [26].

2.5.1 Weighted Sum Scalarisation

When dealing with multiple objectives, the first idea of the decision-maker is often to formulate a new problem with the sum of all objectives, thus creating a scalar-valued optimisation problem. If the resulting solution is not favourable, weights for the individual objectives are introduced. This approach is often taken without explicitly thinking of Pareto-optimality or multi-objective optimisation theory. However, it has also a strong motivation based on the theory of Pareto-optimality. If comparing the multi-objective KKT conditions in Definition 2.37 and those for single-objective optimisation in Definition 2.7 it is clear that the multi-objective conditions are the same as if one were optimising the summed objectives as a scalar optimisation problem. Thus every solution using a weighted sum of objectives satisfies the necessary conditions for Pareto-optimality.
Formally the definition to formulate a single-objective optimisation problem for (MONLP) using the weighted sum scalarisation for $w \in \mathbb{R}^{N_F}$ is

$$\begin{array}{lll} \min\limits_{x \in \mathbb{R}^{N_x}} & \sum\limits_{l=1}^{N_F} w_l F_l(x), & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, \dots, N_g, \\ & h_j(x) = 0, & j = 1, \dots, N_h, \end{array} \tag{2.71}$$

with $\sum_{l=1}^{N_F} w_l = 1$ and $w_l \geq 0$ for $l = 1, \dots, N_F$. However, in practice the weights are chosen as any value greater than zero. This represents a mixture of the weighted sum approach and scaling individual minima such that the theoretical results still apply.

The weighted sum approach is very intuitive and easy to implement. However, it has some drawbacks that shall be addressed at this point. First, the very intuitive property has its limit when adjusting the chosen weights. Das and Dennis [25] show that an equidistant distribution of weights does not lead to an equidistant distribution of samples. Even worse, small changes in

the weights can lead to jumps [25] and at the same time, several different weights can lead to the same optimal solution [43]. In [25] Das and Dennis give a geometrical explanation for this behaviour. A look at the motivation through KKT conditions also reveals that problems formulated by the weighted sum only satisfy the necessary optimality conditions. However, there is no proof that all Pareto-optimal points are optimal solutions for this formulation. On the contrary, Pareto-optimal points in a concave-shaped region of the Pareto front are only saddle points for the problem of type (2.71) [66]. Thus, theoretically, it is impossible to find all Pareto-optimal points via the weighted sum if the respective Pareto front consists of concave parts.
This inability to find all Pareto-optimal points is a criticism often presented against the weighted sum approach. However, throughout the research for this thesis, almost all real-world applications that were investigated did have a convex Pareto front. In addition, the NLP-Solver WORHP on default checks only the first-order conditions. Thus it is still possible to get the saddle-point in the concave regions of the Pareto front as a result of the SQP iterations. However, the problems based on the unintuitive behaviour of the solutions when changing the weights remain. This behaviour is particularly problematic since the weighted sum is often used by people unaware of it.
All in all, the criticism of the weighted sum is justified, but in real-world applications, it should not be rejected outright, mainly because the approach is easy to implement. Some of the problems are also addressed by the adaptive weighted sum approach by Kim and De Weck in [73]. The applicability to trajectory optimisation problems in the context of autonomous ships is investigated in Chapter 5.

2.5.2 ε-Constraint Method

Besides the weighted sum in Section 2.5.1 another simple approach to formulate scalar values optimisation problems is to introduce additional constraints to define an upper limit for all but one objective and optimise for the remaining objective. Similar to the weighted sum, it is very intuitive and widely used [31,32,91]. As the weighted sum, the ε-constraint method is often used by persons who are unfamiliar with multi-objective theory and connections between both are often drawn. For instance, objectives treated within the weighted sum are sometimes referred to as “soft constraints” whereas objectives treated as inequality constraints are also called “hard constraints” in this context. Thus the weighted sum is considered some type of relaxation of constraints originally formulated as inequality constraints. This linguistic connection of the two methods shows their use in optimisation even outside the theory of multi-objective optimisation. In some situations, however, it is advantageous to know about the underlying theory. Especially when the problems under consideration become more complicated and sufficient treatment of the arising problems requires more sophisticated methods.
Formally the ε-constraint method is defined for $m \in \{1, \dots, N_F\}$ and $\varepsilon_l \in \mathbb{R}$ with $l = 1, \dots, N_F$, $l \neq m$ by

$$\begin{array}{lll} \min\limits_{x \in \mathbb{R}^{N_x}} & F_m(x), & \\ \text{subject to} & F_l(x) \leq \varepsilon_l, & l = 1, \dots, N_F,\ l \neq m, \\ & g_i(x) \leq 0, & i = 1, \dots, N_g, \\ & h_j(x) = 0, & j = 1, \dots, N_h. \end{array} \tag{2.72}$$

In contrast to the weighted sum, for every Pareto-optimal point x^* there exists a formulation (2.72) such that x^* is the optimal solution of this problem (see [31] or [91] for a proof). The problem for the applicability of the ε-constraint method is in the other direction. That is that it is not guaranteed that for every combination of $\varepsilon_l \in \mathbb{R}$ there exists a solution to the corresponding

problem (2.72). If the value of a specific ε_l is chosen to be lower than the individual minima of the respective F_l, no feasible solutions exist. Thus the ε-constraint method requires some prior knowledge about the problem. This requirement is one of the main reasons for why a switch to the "soft constraints" is done. At the same time, if the value of ε_l is chosen too high, the respective constraint will become inactive in the solution. Thus the respective objective does not participate in the solution and could also be removed from the objective vector.
All in all, the ε-constraint method also has some drawbacks, mainly that it requires some a-priori knowledge about the possible solutions. Yet, due to its intuitively understandable formulation and easy implementation, it still has its usability.

2.5.3 Pascoletti Serafini Scalarisation

The weighted sum and the ε-constraint methods are probably two of the most used methods, which is likely a result of their intuitive understandable formulation. However, both have some fundamental problems. Hence, a more sophisticated method is advisable when implementing, for example, a general-purpose solver. The scalarisation approach formulated by Pascoletti and Serafini in [98] has not the most intuitive definition. However, it has many favourable properties. For instance, every weakly Pareto-optimal point can be found through this method, and each scalar problem has an optimal solution which is at least weakly Pareto-optimal. The method is very abstract, but provides in this abstractness, a generalisation of the ε-constraint method or the also well-known normal-boundary-intersection method by Das and Dennis [26] which is shown by Eichfelder in [32].
The formal definition of the Pascoletti-Serafini approach is given by

$$\begin{aligned} \min_{x\in\mathbb{R}^{N_x},t\in\mathbb{R}} \quad & t, \\ \text{subject to} \quad & F_l(x) - a_l + tr_l \leq 0, \quad && l = 1,\dots,N_F, \\ & g_i(x) \leq 0, && i = 1,\dots,N_g, \\ & h_j(x) = 0, && j = 1,\dots,N_h, \end{aligned} \tag{PS(a,r)}$$

with $a = (a_1,\dots,a_{N_F})^\top \in \mathbb{R}^{N_F}$, $r = (r_1,\dots,r_{N_F})^\top \in \mathbb{R}^{N_F}$ and $r_l > 0$, for all $l = 1,\dots,N_F$. The formulation can also be adapted if optimality is not limited to Pareto-optimality but expressed via more general cone based orderings. Eichfelder explains this in the book [32] and also provides more details about the theoretical properties.

Solving problem (PS(a,r)) also has a geometrical interpretation. The idea is to go from a reference point a along the line tr until the intersection of the set $\{x \mid F_l(x) \leq a_l + tr_l \leq 0,\ l = 1,\dots,N_F\}$ and the feasible set Σ (2.1) is the empty set. The minimal value of t for which the set is not empty is the optimal value t^* and x^* is the point x which is in the intersection set. Figure 2.3 sketches the geometric interpretation with the setting in the optimal point (x^*, t^*) in red, the feasible set in blue and the different parts of the additional constraint $F_l(x) - a_l + tr_l \leq 0$, $l = 1,\dots,N_F$ in black, green and magenta.

Eichfelder [32] shows that every solution of (PS(a,r)) is a weakly Pareto-optimal point. Theorem 2.38 formally defines this property.

Theorem 2.38
Let (x^*, t^*) be a local minimal solution of (PS(a,r)), then x^* is a locally weakly Pareto-optimal solution of the multi-objective optimisation problem (MONLP).

Proof: See [32] □

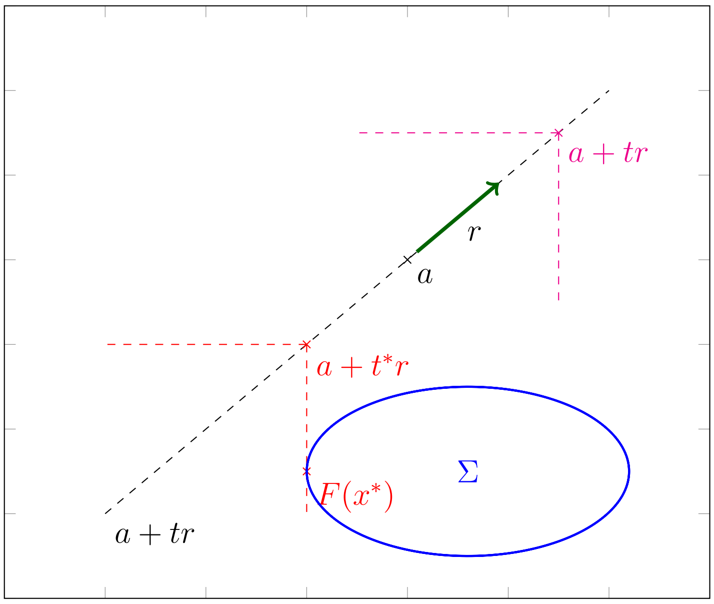


Figure 2.3: Geometric description of the Pascoletti-Serafini scalarisation.

The other direction that every weakly Pareto-optimal point can be computed as the solution of (PS(a, r)) for a specific parameter set (a, r) is also true.

Theorem 2.39
Let x^*be a weakly Pareto-optimal solution of the multi-objective optimisation problem (MONLP), then $(x^*, 0)$ is a minimal solution of (PS(a, r)) for the parameter $a := F(x^*)$ and for arbitrary $r \in \mathbb{R}^{N_F} : r_l > 0,\ l = 1, \ldots, N_F$.

Proof: See [32] □

For the theorem, that Problem (PS(a, r)) always has a solution if the respective multi-objective optimisation problem (MONLP) has Pareto-optimal points [32], the choice of r is important. If for example $r_l = 0$ for some $l = 1, \ldots, N_F$ it is not guaranteed that (PS(a, r)) has a solution for every choice of $a \in \mathbb{R}^{N_F}$. Interestingly, to define the ε-constraint method by means of the Pascoletti-Serafini scalarisation the parameter r has to be chosen such that it contains zero entries (see Eichfelder [32]) thus violating this condition.

So far Problem (PS(a, r)) depends on two hyperparameters, a and r. However, Eichfelder [32] also shows that, to find all Pareto-optimal solutions for a given problem (MONLP), it is sufficient to choose r to be constant and to vary a along a $(N_F - 1)$-dimensional hyperplane. This fits with the interpretation in Hillermeyer [66] that the Pareto front is a locally $N_F - 1$ dimensional manifold since there exists a map from an $N_F - 1$ dimensional space into the objective space which represents the Pareto front. This map is important in Chapter 3 where it is interpolated, based on parametric sensitivity information.
Choosing the parameter a from within a hyperplane still does not limit the parameter. Thus, even when choosing only from a discretised set of the hyperplane, an infinite number of scalarisations have to be computed. To restrict the hyperplane, there exist different approaches. A

simple one is to choose the simplex spanned by the individual minima as proposed for the Normal-Boundary-Intersection method in [26]. This set is easy to compute and does not waste any computational power since the individual minima are also Pareto-optimal and can thus be used within the approximation of the Pareto front. However, for $N_F \geq 3$, this method usually does not yield all Pareto-optimal points since the hyperplane is restricted too closely.
Eichfelder [32] proposes a method where a set of other optimisation problems is solved to find an outer approximation of the Pareto front, similar to a reachable set algorithm. This method will still yield parameters a, which yield the same solution x^*. This is the case if the line $a + tr$ has no intersection with the Pareto front, leading to one of the additional constraints being inactive. This situation is shown in Figure 2.3 where the line does not intersect the feasible set Σ. If the parameter a is varied sufficiently small in this situation, the same solution x^* will be computed. This might waste computational power. However, Eichfelder [32] also shows how a can be adapted when not all constraints are active to formulate a Problem ($\text{PS}(a, r)$) with all constraints active.
The next value for a can then be chosen from this new position. Thus it can be prevented to find the same solution too often. This method requires some implementation effort, especially in higher dimensions. In addition, the initial optimisation problems used to find the outer approximation have shown to be numerically harder to solve, resulting in a higher number of iterations to finish. Also, their solutions are not Pareto-optimal and can thus not be used to approximate the Pareto front. In this thesis, the simplex between individual minima is used as the set from which to choose a. Since most considered problems have the dimension $N_F = 2$, there is no problem regarding the completeness of the approximation. If $N_F \geq 3$, the samples computed based on the simplex are often the most interesting ([26]). It is also a possibility to start with the approximation based on the simplex between individual minima, and then choose a new parameter one discretisation step out of the hyperplane used so far until the border is reached, which is detected by an inactive constraint.

All in all, the Pascoletti-Serafini approximation yields several favourable properties for real-world applications. The main disadvantage compared to the weighted sum is the more complicated formulation. In Chapter 5 both methods will be compared with respect to their performance in an example from autonomous ship control.
Besides the general properties explained in this section, the Pascoletti-Serafini method also has the advantage that the Lagrange multiplier for the additional constraints represents the tradeoffs between the individual objectives [61]. This feature will be investigated in Chapter 3 where it is first extended via the general parametric sensitivity theory and afterwards used to compute an interpolation of the Pareto front.

Chapter 3

Using Parametric Sensitivity Analysis Information for better Pareto Front Approximations

Contents

For an objective function $F : \mathbb{R}^{N_x} \to \mathbb{R}^{N_F}$ the Pareto front can be described locally as a $N_F - 1$ dimensional manifold in the objective space if both the objective functions and the constraints are twice continuously differentiable (see Hillermeier [66]). Within this chapter, this fact is exploited in the way that the maps $\mathcal{M} : \mathbb{R}^{N_F-1} \to \mathbb{R}^{N_F}$ which describe the manifold locally can be approximated. This approximation, in turn, provides the decision-maker with an approximation of the Pareto front itself.
The approximation is made by computing a single sample of the Pareto front via the Pascoletti-Serafini (PS(a, r)) scalarisation scheme with the hyperparameter $a, r \in \mathbb{R}^{N_F}$. Subsequently, the parametric sensitivity information (see Chapter 2.2) is computed for this sample. This information provides derivatives of the maps $\mathcal{M}$, enabling a Taylor-Series development of these maps.
In general, the idea of using derivative information of the Pareto front manifold is not new. The first-order derivatives are usually considered as the local tradeoffs between individual objectives. Haimes and Chankong describe the connection between these tradeoffs and the Lagrange multiplier in [20, 61]. Also, using the first-order derivatives to approximate the Pareto front linearly is already proposed, for example, as within the adaptive scalarisation method described by Eichfelder [33].

This thesis focuses on the more general approach via parametric sensitivity analysis. Through this very general theory, it is possible to exploit more sensitivities than only those of the objectives themselves. Second-order derivatives and those of the variables are used to provide higher-order and different approximations, respectively. It is also investigated if and how the local information from the parametric sensitivity analysis can be used to approximate the Pareto front between two different samples as one single continuous or even differentiable function.

In this chapter, the local approximation of the Pareto front is first described and investigated for bi-objective examples. Thereafter, it is shown how the approximations can be used globally between two different Pareto front samples. It is investigated how this can reduce the number of samples needed to get a good approximation of the Pareto front. The most important limitations and problems of this approach are shown afterwards. Finally, it is discussed how the methods can be extended to dimensions $N_F > 2$. For this, a small example is used for the local approximation. For the investigations in this chapter the following four examples are used.

Example 3.1 (ArcTan)
This example consisting of a trivial objective function and one simple but nonlinear non-polynomial constraint is used to show basic features of the investigated methods:

$$\begin{aligned} \min_{x\in\mathbb{R}^2} \quad & (x_1, x_2)^T, \\ \text{subject to} \quad & x_1 = \operatorname{atan}(-x_2). \end{aligned} \tag{3.1}$$

Example 3.2 (NBI)
This well known example from [26] is used to demonstrate features when the objective function is not as trivial as in Example 3.1:

$$\begin{aligned} \min_{x\in\mathbb{R}^5} \quad & \begin{pmatrix} x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \\ 3x_1 + 2x_2 - \frac{x_3}{3} + 0.01(x_4 - x_5)^3 \end{pmatrix}, \\ \text{subject to} \quad & x_1 + 2x_2 - x_3 - 0.5x_4 + x_5 = 2, \\ & 4x_1 - 2x_2 + 0.8x_3 + 0.6x_4 + 0.5x_5^2 = 0, \\ & x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \le 10. \end{aligned} \tag{3.2}$$

Example 3.3 (Rocket Car)
This example is derived from the discretisation of an optimal control problem which describes the optimisation of a "Rocket Car" which can only accelerate. The objective functions represent the maximisation of the driven distance χ_1 and the minimisation of the consumed energy χ_2. It is discretised via direct methods using the trapezoidal scheme (see [8]) resulting in

$$\begin{aligned} \min_{\tilde{x}\in\mathbb{R}^{(2)(N_t+1)},\ \tilde{u}\in\mathbb{R}^{(1)(N_t+1)}} \quad & \begin{pmatrix} -\chi_1^{[N_t]} \\ \chi_2^{[N_t]} \end{pmatrix}, \\ \text{subject to} \quad & \chi^{[i]} = \chi^{[i-1]} + \frac{h}{2}\left(f(\chi^{[i-1]}, u^{[i-1]}) + f(\chi^{[i]}, u^{[i]}) \right), \quad i = 1, \ldots, N_t, \\ & \chi^{[0]} = (0,0)^\top, \\ & 0 \le u_1^{[j]} \le 2, \quad j = 0, \ldots, N_t, \end{aligned} \tag{3.3}$$

with

$$f(\chi, u) = (\chi_2, u_1)^\top,\ N_t = 11 \text{ and } h = 10^{-6}. \tag{3.4}$$

Here $\chi^{[i]}$ and $u^{[i]}$ represent the states and controls at the ith discrete point of an discretised optimal control problem. The optimisation variables $\tilde{x}$ and $\tilde{u}$ are a concatenation of all discrete points. This example is used to demonstrate certain problematic behaviours especially with bang-bang control problems which result from the linear influence of all controls in this problem.

Example 3.4 (Thomann T8 (P.38))
This example is taken from [114] and is used to demonstrate methods for dimensions $N_F > 2$:

$$
\begin{aligned}
\min_{x \in \mathbb{R}^3} \quad & \begin{pmatrix} \sum_{l=1}^{3} x_l^3 \\ \sum_{l=1}^{2} (x_l - 4)^2 + x_3^2 \\ -\ln(x_1) + 5\sum_{l=2}^{3} x_l^2 \end{pmatrix}, \\
\text{subject to} \quad & 0 < x_1 \leq 10, \\
& 0 \leq x_2 \leq 10, \\
& 0 \leq x_3 \leq 10.
\end{aligned} \tag{3.5}
$$

Note 3.5
The examples used in this chapter were purposefully picked to demonstrate certain behaviours of the described methods and should not be considered a test set for benchmarks in any kind.

3.1 Using Parametric Sensitivity Information for Local Approximation of the Pareto Front in Bi-Objective Optimisation

This section shows how the information obtained via parametric sensitivity analysis can be used to approximate the Pareto front locally around a sample computed with the Pascoletti-Serafini scalarisation scheme. For this, the sensitivities of the first and second order of the individual objective functions are used as well as the sensitivities of the variables.

3.1.1 Using Parametric Sensitivity Information of the Objectives for Local Approximation

As mentioned before, the Pareto front is locally a $N_F - 1$ dimensional manifold in the objective space. This fact yields the existence of maps $\mathcal{M} : \mathbb{R}^{N_F - 1} \to \mathbb{R}^{N_F}$. The idea presented in this chapter is to find local approximations of these maps around a sample computed with the Pascoletti-Serafini scalarisation. For an introduction of the Pascoletti-Serafini approach see Section 2.5.3.
When approximating the Pareto front with the Pascoletti-Serafini scalarisation scheme, one usually varies the hyperparameter $a \in \mathbb{R}^{N_F}$ along a $\mathbb{R}^{N_F - 1}$-dimensional hyperplane. Thus one idea would be to describe the maps $\mathcal{M}$ via a dependency of a as $\tilde{\mathcal{M}}(a)$. Since a itself is an element of $\mathbb{R}^{N_F}$ it is then necessary to describe a as function $a(\tau)$ with $\tau \in \mathbb{R}^{N_F - 1}$. This is possible by describing it as a linear function since a is varied along a linear hyperplane. All in all, this yields the chain of functions $\tilde{\mathcal{M}}(a(\tau)) : \mathbb{R}^{N_F - 1} \to \mathbb{R}^{N_F}$.
Now, the map $\mathcal{M}$ can be defined by

$$\mathcal{M}(\tau) := \tilde{\mathcal{M}}(a(\tau)) := F(x^*(a(\tau))) \text{ where } x^* = \text{ argmin PS}(a(\tau), r), \tag{3.6}$$

for a given $r \in \mathbb{R}^{N_F}$ as described in Section 2.5.3.

The goal is to approximate $\mathcal{M}$ as best as possible, usually done by setting different values for τ and solving Problem $\text{PS}(a(\tau), r)$ respectively. Thus, computing these samples leads to the necessity of solving lots of nonlinear optimisation problems. This chapter focusses on an idea about how to get more information out of each sample so that fewer samples are needed in total, and the graph of $\mathcal{M}$ and thus the Pareto front can be better visualised.

This is achieved through defining several different local approximations via Taylor series developments of $F(x^*(a(\tau)))$. This idea is not new, Pascoletti and Serafini [98] already discussed how the solution F^* changes locally when a and r are varied. Eichfelder [33] also discusses this idea for different preconditions and for the general case where Pareto-optimality is defined through a cone C. The case $C = \mathbb{R}_+^{N_F}$ as the part of $\mathbb{R}^{N_F}$ with only non-negative components is equivalent to the definition of Pareto-optimality used in this thesis.

Within this thesis, these ideas are used and extended. The main difference is that the local behaviour of $F(x^*(a(\tau)))$ for small changes of τ is not investigated by means specific for the Pascoletti-Serafini method but described through the results from the general parametric sensitivity analysis from Chapter 2.2. This analysis enables second-order approximations of $\mathcal{M}$ by using the sensitivity information of the Lagrangian multipliers. Also, the approximation of the variables $x \in \mathbb{R}^{N_x}$ of the problem is thus possible. The latter yields also an approximation of the Pareto front when F is applied to the approximated variables. The techniques described will be shown for the bi-objective case $N_F = 2$ first. How it can be generalised will be discussed at the end of this chapter. To the author's knowledge, this more general approach is a new viewpoint of possible approximations, which leads to new ideas and tools to provide as much information to the decision-maker as possible.

First, this approach will be shown for the first and second derivatives of $\mathcal{M}$. It will be developed how the information can be obtained from the parametric sensitivity information. It will then be demonstrated with a bi-objective example how this information can help to extend the knowledge about the shape of the Pareto front without computing new samples for the approximation.
Second, it is discussed if an application of F to approximated variables can also help in understanding the Pareto front and how it compares to the other approximations.

In order to compute a Taylor-series development of $\mathcal{M}$, the function is broken down into the approximation of the behaviour of $F(x^*(a))$ when a changes and subsequently combined with the change of a with respect to τ. For this, it is necessary to understand how the function changes locally when a is varied locally around a reference point. Since $F(x^*(a))$ is defined through the optimal solution of $(\mathrm{PS}(a,r))$ for a specific a the question is how this optimal solution changes with a change of a. However, this question of change is exactly what parametric sensitivities can answer for any optimisation problem that satisfies the necessary preconditions. Thus, it is only necessary to show the connection between the parametric sensitivities of a for $(\mathrm{PS}(a,r))$ and the behaviour of F, which is not directly visible since F acts as part of the additional constraints in $(\mathrm{PS}(a,r))$.
Eichfelder [33] already showed how the parametric sensitivity information of the scalarised bi-objective problem $(\mathrm{PS}(a,r))$ can be used to get the derivative of each objective function F_l:

$$\nabla_a F_l = {e^{[l]}}^\top + \begin{pmatrix} \frac{\partial t}{\partial a_1} & \frac{\partial t}{\partial a_2} \end{pmatrix} r_l = {e^{[l]}}^\top + \begin{pmatrix} \lambda_{\hat{l}_1} & \lambda_{\hat{l}_2} \end{pmatrix} r_l, \quad l = 1,2, \tag{3.7}$$

with $e^{[l]}$ as the vector with all zeros except for the lth entry which is 1.
The last equality originates in the fact that

$$\frac{\partial t}{\partial a_l} = \lambda_{\hat{l}_l}, \quad l = 1,2, \tag{3.8}$$

with $\hat{l}_l := N_g + N_h + l$, which was also shown in [33] but can alternatively be obtained from the general parametric sensitivity analyses as described in [18].
By computing (3.7) it becomes possible to do a first-order Taylor series development for the functions F_1 and F_2 dependent on a and with the knowledge that a is a linear function of τ it is also possible to compute an approximation of F with regard to τ. With $\Delta a = (a(\tau) - a(\tau_0))$ and $\Delta\tau = \tau - \tau_0$ the approximation becomes

$$F_l(x^*(a(\tau))) \approx F_l(x^*(a(\tau_0))) + \nabla_a F_l(x^*(a(\tau_0)))\Delta a \Delta\tau, \quad l = 1, 2. \tag{3.9}$$

Also, since parametric sensitivities capture all the implicit dependencies in the problem, this approximates $\mathcal{M}$. So far, this is already known from [61] and Eichfelder describes in [33] how this linearisation of the Pareto front can be used in an adaptive scalarisation algorithm.

However, this approach has not yet been extended with information from the parametric sensitivity information in general. This extension will be done in the following. To be able to form a second order Taylor series development of (3.6) the second derivative of the function $F(x^*(a))$ and, for this, of the objective function F is needed. Combining the fact that the parametric sensitivity analysis provides the derivatives of the Lagrangian multipliers λ for all possible perturbations with the correlation (3.8), it follows that

$$\nabla_a^2 F_l = \begin{pmatrix} \frac{\partial^2 F_l}{\partial a_1^2} & \frac{\partial^2 F_l}{\partial a_1 \partial a_2} \\ \frac{\partial^2 F_l}{\partial a_2 \partial a_1} & \frac{\partial^2 F_l}{\partial a_2^2} \end{pmatrix} r_l = \begin{pmatrix} \frac{\partial \lambda_{\hat{l}_1}}{\partial a_1} & \frac{\partial \lambda_{\hat{l}_1}}{\partial a_2} \\ \frac{\partial \lambda_{\hat{l}_2}}{\partial a_1} & \frac{\partial \lambda_{\hat{l}_2}}{\partial a_2} \end{pmatrix} r_l, \quad l = 1, 2. \tag{3.10}$$

This, in turn, makes it possible to formulate the second derivative of $\tilde{\mathcal{M}}(a(\tau))$ with respect to $a(\tau)$

$$\left[\nabla_a^2 \tilde{\mathcal{M}}_l\right] = a(\tau)^\top \; \nabla_a^2 F_l \; a(\tau), \quad l = 1, 2, \tag{3.11}$$

which again can be used to form the second order Taylor series development of (3.6) as

$$\begin{aligned} F_l(x^*(a(\tau))) \quad \approx \quad & F_l(x^*(a(\tau_0))) + \nabla_a F_l(x^*(a(\tau_0)))\,\Delta a \Delta\tau \\ & + 0.5\,\Delta a^\top\, \nabla_a^2 F_l(x^*(a(\tau_0)))\,\Delta a \Delta\tau^2, \qquad l = 1, 2. \end{aligned} \tag{3.12}$$

The potential of these approximations for visualising Pareto fronts accurately with a low number of samples will be demonstrated in the following by using Example 3.1. This example was purposefully selected because it shows the potential of the method. Possible problems and pitfalls are later discussed using the Example 3.2.

Note 3.6
Eichfelder [33] uses the modified Pascoletti-Serafini scalarisation to prove the behaviour of $\mathcal{M}$ around a given value of a. The modification is to replace the inequality of the additional constraints in (PS(a, r)) with equality. So far, this was not mentioned with regards to the motivation through parametric sensitivities in general. However, parametric sensitivities are only valid for active constraints, and since all sensitivities of the additional constraints are used, it is equivalent to the assumption that equality was used.
Eichfelder also describes that for each solution for (PS(a, r)) there exists an a such that the equality holds. How this a can be computed is also described by Eichfelder. If this approach is

to be used in real applications, it is necessary to check if all constraints are active and recompute the solution with an appropriate a if necessary. However, since the optimal solution is then known, except for the dual variables, this recomputation should usually be much faster than solving the problem in general.

For the results shown in this chapter, it was checked if the constraints are active. Numerical experiments also showed that this precondition is a theoretical one needed for the formal proof, but that the methods described fail to give any valuable information when the condition is not satisfied.

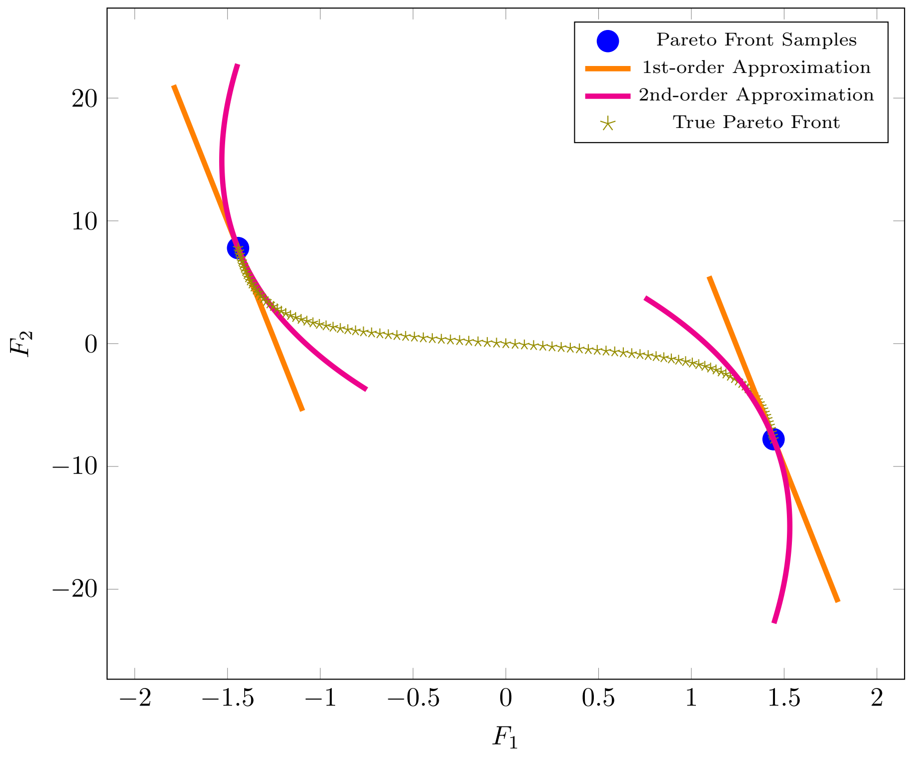


Figure 3.1: Local Pareto front approximations of 1st- and 2nd-order for Example 3.1, compared to true Pareto front.

In Figure 3.1 the approximations of the 1st- and 2nd-order are shown for Example 3.1. To compare the approximation with the true Pareto front, it was sampled with a higher resolution and the resulting samples are displayed as olive stars in Figure 3.1.
The example displays the advantage of additional information obtained via parametric sensitivity analysis. With the classical approach, where only the samples are computed, it is not possible to get an insight into the shape of the Pareto front between two samples. Would the samples in Figure 3.1 be connected with a straight line, it would result in a completely wrong idea of the Pareto front. This example is also constructed in a way where even a third point in the middle would not yield any more information about the shape if only the samples themselves were considered. Thus, at least four samples, which is twice the number shown in Figure 3.1, are needed to get a rough overview.

By utilising the fact that the Lagrangian multipliers λ represent a linear sensitivity as described in [33], it becomes possible to compute and display an approximation done by a first-order Taylor series development. The result for both samples is displayed in Figure 3.1 by the orange lines. By using this knowledge, it is already apparent that a straight line connecting both samples is not the best way to approximate the Pareto front. Furthermore, the approximation gives information about the local tradeoffs between the single-objectives.

Through the additional use of parametric sensitivity information of λ, as proposed in this thesis, it is also possible to display the curvature around the computed samples. This is shown as magenta lines in Figure 3.1. When considering only one example at a time, this approach makes it possible to consider tradeoffs and reveals directly if a given sample is in a convex or a concave part of the Pareto front. Furthermore, the approach shows how the tradeoffs would change when going in a certain direction. Also, when displayed as in Figure 4.3 it provides an excellent idea of the overall shape of the Pareto front.

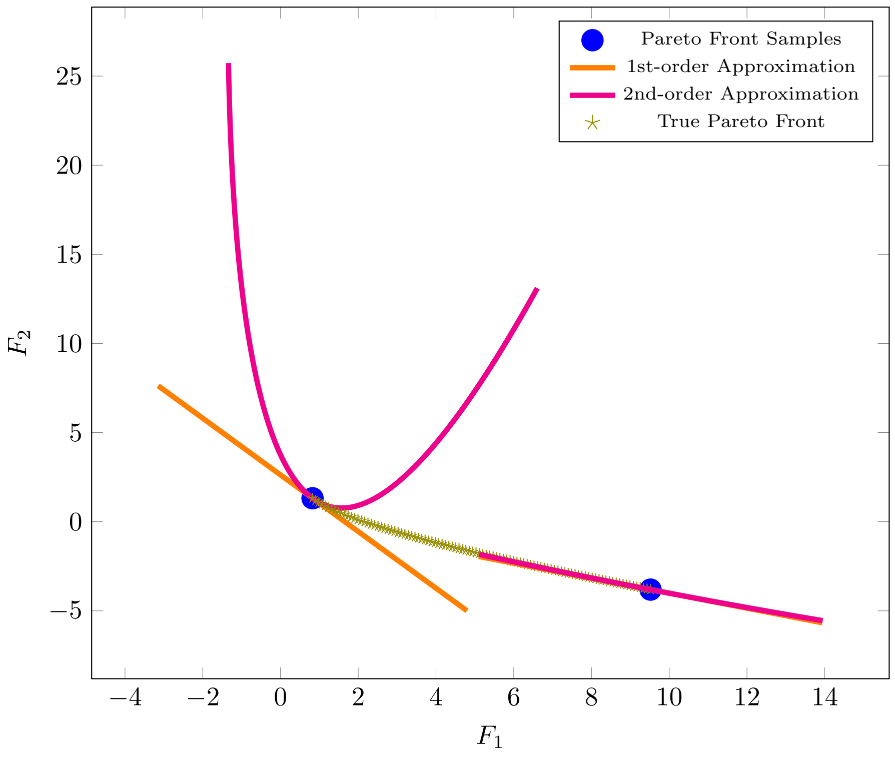


Figure 3.2: Local Pareto front approximations of 1st- and 2nd-order for Example 3.2, compared to true Pareto front.

Example 3.1 and the results in Figure 3.1 display a very good behaviour regarding the proposed methods. To show that this is not always the case, another example is investigated at this point. Example 3.2 is a well known example from the paper proposing the Normal Boundary Intersection method [26]. Figure 3.2 shows the results for Example 3.2 obtained with the same methodology as for Example 3.1 in Figure 3.1.
This example shows a very different behaviour at each of the two samples. First, the two approx-

imations around the right sample at $F_1 \approx 10$ are the same and both approximate the true Pareto front almost perfectly. The reason for this is that the Pareto front at this point has an almost linear graph which leads to an excellent linear approximation and the fact that a second-order approximation does not directly add much new information. However, the information that both approximations are almost identical can be valuable in itself, as it shows that it might not be necessary to compute more samples in this area. The example also shows that the second-order approximation does not destroy an excellent linear approximation when the second derivatives obtain no additional information.
The second sample around $F_1 \approx 1$ displays quite a different behaviour. Here, the linear and quadratic approximations do not fit the true Pareto front and differ a lot from each other. However, regarding the other sample, the comparison of linear and quadratic approximation provides a piece of information in itself, namely that at this point, the shape of the Pareto front might not be trivial. There is also the implicit information that the second-order approximation is not sufficient, which can be obtained by the fact, that the approximation function is dominated by itself in the area $F_1 > 1$. Thus, if it is assumed here that the Pareto front is connected, it is clear that the approximation does not display the shape of the Pareto front. If that assumption can not be made, it might also hint that the sample is close to a discontinuity of the Pareto front. All in all, it can be stated that the first and second-order approximations add valuable information for the decision-maker.

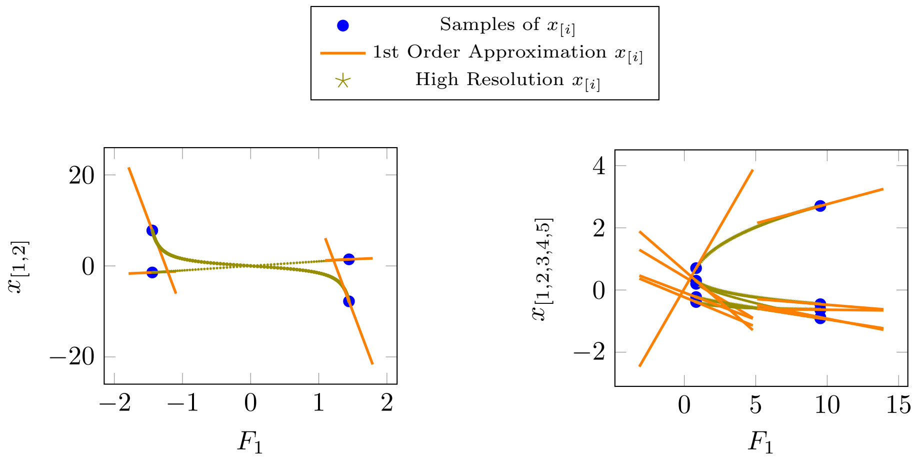
The examples show how parametric sensitivity analysis can extend the knowledge that can be obtained from a single sample of the Pareto front. It should also be noted that this information requires no or only minimal additional computational effort. This is because parametric sensitivities can be computed efficiently. Thus, the 1st-order information is directly available, given by the Lagrangian multipliers λ. Additionally, the parametric sensitivity of λ can also be computed efficiently by reusing matrix factorisations already computed while solving the scalarised problem. See [80] and [18] for more details on this topic.

3.1.2 Using Parametric Sensitivity Information of the Variables for Local Approximation

In Section 3.1.1 it was described how the parametric sensitivity of the Lagrangian multiplier can be used to extend the knowledge about the local shape of the Pareto front. This section will show how the parametric sensitivity information of the variables x for the hyperparameter a in (PS(a, r)) can also be utilised for another approach of obtaining more insight about the Pareto front.

By considering x analogously to (3.6) as an implicitly defined function $x(a(\tau))$ it becomes clear that it should also be possible to form a first-order approximation of this function. Therefore, the theory of parametric sensitivity analysis provides derivative information of the objective function and can be used to compute the first derivative of the variables for a given perturbation parameter. Thus, by considering the hyperparameter a as a perturbation, as was already done in Section 3.1.1, it is possible to compute the value of $\frac{\partial x}{\partial a}(a(\tau))$, which again can be used to do a Taylor series development for x_i.

The approximation of the variables is shown in Figure 3.3a for Example 3.1 and in Figure 3.3b for Example 3.2. To better show connections between the graphs of the variables and those of the objective functions in Figures 3.1 and 3.3b the variables of the optimal solutions are plotted



(a) Example 3.1

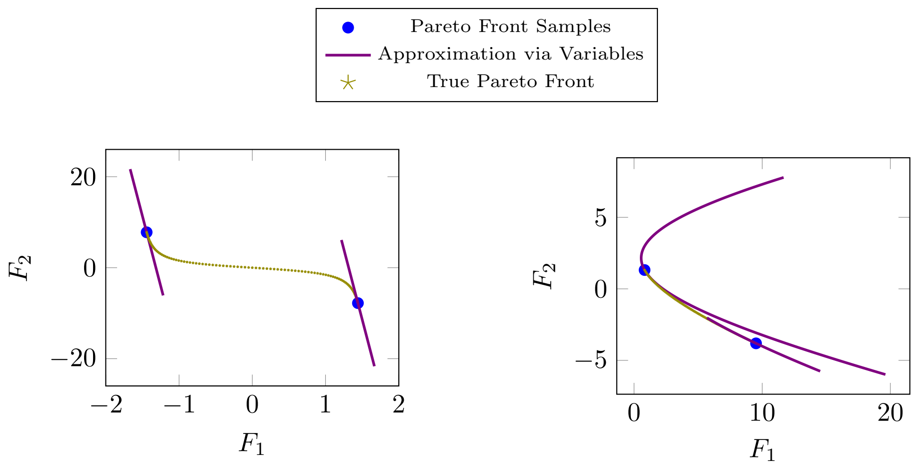
(b) Example 3.2.

Figure 3.3: Local variable approximations of 1st-order for variables x_i, $i = 1, \ldots, N_x$.

against the first objective function F_1. In this way, the allocation of the variable samples to those of the objectives becomes very intuitive.
In the example shown in Figure 3.3a it is displayed that x_2 has almost the same graph as the objective functions in Figure 3.1 and x_1 is linear between -1 and 1. Since the objective function in Example 3.1 are identity functions for each dimension, it is clear that the variables are shaped analogously to the Pareto front itself. The plots for Example 3.2 in Figure 3.3b show no special behaviour. These Examples show mainly that the linear approximation of the variables works in general. However, since in reality the value of N_x can reach up to 10^9, it is also clear that visualising the variables, in general, might not be practical unless some select few variables are of special interest.
The idea proposed here is not to evaluate approximate and display the variables themselves but to compute values for the objective function at the approximated variables $\tilde{x}$, thus obtaining one more approximation for the objective function as $F(\tilde{x})$. In this way, even though the approximation $\tilde{x}$ of x is linear, it would be possible to preserve the nonlinearity introduced by the objective functions.

Results of this approach for the Examples 3.1 and 3.2 are shown in Figures 3.4a and 3.4b respectively. First, it can be directly seen that for Example 3.1 this approach does not differ from the linear approximation done directly for the objective functions which is shown in Figure 3.1. Since the objective functions in this example are identity functions, this is the expected behaviour. However, it supports the view that this approach might preserve the nonlinearities by displaying that if there are no nonlinearities it is identical.

The graphs displayed for Example 3.2 in Figure 3.4b show a different behaviour. Especially the approximation around the left sample shows how a linear approximation of the variables can still be used to get a nonlinear approximation of the Pareto front which, in this case, is already an excellent approximation. Compared to all approximations shown in the last section in Figure 3.2 this one shows the most correct graph. In addition it hints at where the high curvature in the second-order approximation in Figure 3.2 originates from.



(a) Example 3.1.

(b) Example 3.2.

Figure 3.4: Local Pareto front approximations by 1st-order of variables.

The high quality of the approximations in Figure 3.4b are a bit surprising in the way that the linear approximations of the variables shown in Figure 3.3b are not nearly as accurate as would be expected. However, there is one inaccuracy that might originate in the inaccurate approximation of the variables, namely that the graph of the approximation of the left sample in Figure 3.4b goes way beyond the second sample on the right. This progression of the graph is not correct since the functions were evaluated for values of τ that should result in samples between the two that have already been computed.

One disadvantage of this approach is that the tradeoffs between the objective functions are not explicitly computed. However, since the approaches can be used complementary and the tradeoffs are directly available with the Lagrangian multipliers, this is no major problem. Alternatively, it is also possible to compute the partial derivatives of the objective functions $\frac{\partial F_l}{\partial x_i}$ at the computed variable values.

Last, it should be noted that even when the primary goal of this approach is another Pareto front approximation, the approximations of the variables are also very useful. In contrast to all other approaches, this can provide correlating variables for every approximated point in the objective space. It is also independent of the choice of scalarisation approach.

3.1.3 Conclusion of the Multiple Approaches for a Local Approximation of the Pareto Front

In Sections 3.1.1 and 3.1.2 multiple approaches were discussed to extend the information obtained by one sample of the Pareto front. It was shown how the information about tradeoffs between objectives could be obtained via parametric sensitivity analysis. Subsequently, this information was extended with the parametric sensitivities of the Lagrangian multiplier λ, thus representing derivatives of second order for the Pareto front. It was shown how these could be used to approximate the Pareto front locally with high accuracy.
Then it was discussed how the sensitivities of the variables could also be used to generate another

approximation of the Pareto front. It was shown that this approach could map the nonlinearities of the objective functions to the approximation, even though the variables are only approximated linearly.

The following section will extend the discussed approaches, which combine the local approximations around the samples to obtain an approximation that is continuous between all computed samples.

3.2 Using Parametric Sensitivity Information for Global Approximation of the Pareto Front in Bi-Objective Optimisation

In Section 3.1 parametric sensitivity information is used to build an approximation of the Pareto front locally around the computed samples. This section discusses the possibility to combine information from neighbouring samples $x^{*[0]} = x^*(a(\tau^{[0]}))$ and $x^{*[1]} = x^*(a(\tau^{[1]}))$ to construct a continuous approximation for the complete interval between both samples.

$F(x^{*[1]}),\ F(x^{*[2]})$	1st grade Polynomial
$\frac{\partial F_l}{\partial \tau}\left(x^{*[0]}\right),\ \frac{\partial F_l}{\partial \tau}\left(x^{*[1]}\right)$	3rd grade Polynomial
$\frac{\partial F_l}{\partial \tau}\left(x^{*[0]}\right),\ \frac{\partial F_l}{\partial \tau}\left(x^{*[1]}\right),\ \frac{\partial^2 F_l}{\partial \tau^2}\left(x^{*[0]}\right),\ \frac{\partial^2 F_l}{\partial \tau^2}\left(x^{*[1]}\right)$	5th grade Polynomial
$x^{*[0]},\ x^{*[1]},\ \frac{\partial x}{\partial \tau}\left(x^{*[0]}\right),\ \frac{\partial x}{\partial \tau}\left(x^{*[1]}\right)$	$F($ 3rd grade Polynomial $)$

Table 3.1: Possible combinations of information to form a polynomial approximation between samples.

The proposed idea is to take the information obtained by a sample and the respective sensitivity analysis and compute a polynomial function corresponding to values and derivatives at the given samples. The possible combinations in Table 3.1 show that with the parametric sensitivity information, it is possible to find polynomials with a grade of up to 5 in the form

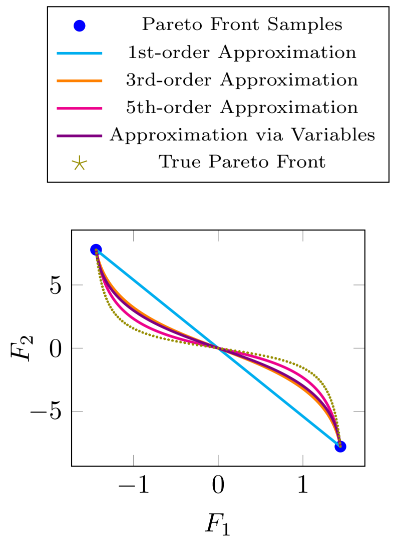
$$F_l\left(x(a(\tau))\right) \approx \rho_5\tau^5 + \rho_4\tau^4 + \rho_3\tau^3 + \rho_2\tau^2 + \rho_1\tau + \rho_0, \tag{3.13}$$

by solving the equation system

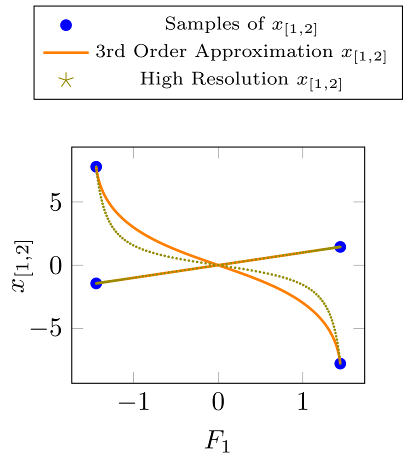
$$\begin{pmatrix} 1.0 & \tau^{[0]} & (\tau^{[0]})^2 & (\tau^{[0]})^3 & (\tau^{[0]})^4 & (\tau^{[0]})^5 \\ 1.0 & \tau^{[1]} & (\tau^{[1]})^2 & (\tau^{[1]})^3 & (\tau^{[1]})^4 & (\tau^{[1]})^5 \\ 0.0 & 1.0 & 2\tau^{[0]} & 3(\tau^{[0]})^2 & 4(\tau^{[0]})^3 & 5(\tau^{[0]})^4 \\ 0.0 & 1.0 & 2\tau^{[1]} & 3(\tau^{[1]})^2 & 4(\tau^{[1]})^3 & 5(\tau^{[1]})^4 \\ 0.0 & 0.0 & 2.0 & 6(\tau^{[0]}) & 12(\tau^{[0]})^2 & 20(\tau^{[0]})^3 \\ 0.0 & 0.0 & 2.0 & 6(\tau^{[1]}) & 12(\tau^{[1]})^2 & 20(\tau^{[0]})^3 \end{pmatrix} \begin{pmatrix} \rho_0 \\ \rho_1 \\ \rho_2 \\ \rho_3 \\ \rho_4 \\ \rho_5 \end{pmatrix} = \begin{pmatrix} F_l\left(x(a(\tau^{[0]})), p_0\right) \\ F_l\left(x(a(\tau^{[1]})), p_0\right) \\ \frac{\partial F_l}{\partial \tau}\left(x(a(\tau^{[0]})), p_0\right) \\ \frac{\partial F_l}{\partial \tau}\left(x(a(\tau^{[1]})), p_0\right) \\ \frac{\partial^2 F_l}{\partial \tau^2}\left(x(a(\tau^{[0]})), p_0\right) \\ \frac{\partial^2 F_l}{\partial \tau^2}\left(x(a(\tau^{[1]})), p_0\right) \end{pmatrix}, \tag{3.14}$$

or parts of it respectively, if less information is taken into account. The polynomials for the approximation via the variables are done equivalently.
Since those polynomials are defined by first and second derivative information at the sample points, the "smoothness" is respectively high, leading to functions that are up to twice continuously differentiable at the connection points when concatenated for the several samples over the

Pareto front.
There are two general types of approximations investigated at this point. The first one corresponds to the results in Section 3.1.1 where the objective functions were directly approximated locally. The first three possible combinations in Table 3.1 represent this approach with different orders of polynomial approximations. Within these three possibilities, the first one where a linear polynomial between the samples is computed, represents the classical approach of just connecting the points. One could also say that this is the natural way a human would connect the samples when presented with the visual representation of a sampled Pareto front.
The last combination in Table 3.1 corresponds to the approach in Section 3.1.2. Here, the variables are approximated based on their values and parametric sensitivities. Afterwards, the objective function is applied to the approximated variables.



(a) Global Pareto front approximations of 1st-, 3rd- and 5th-order.



(b) Global variable approximations of 3rd-order.

Figure 3.5: Global approximations of different orders orders for Example 3.1.

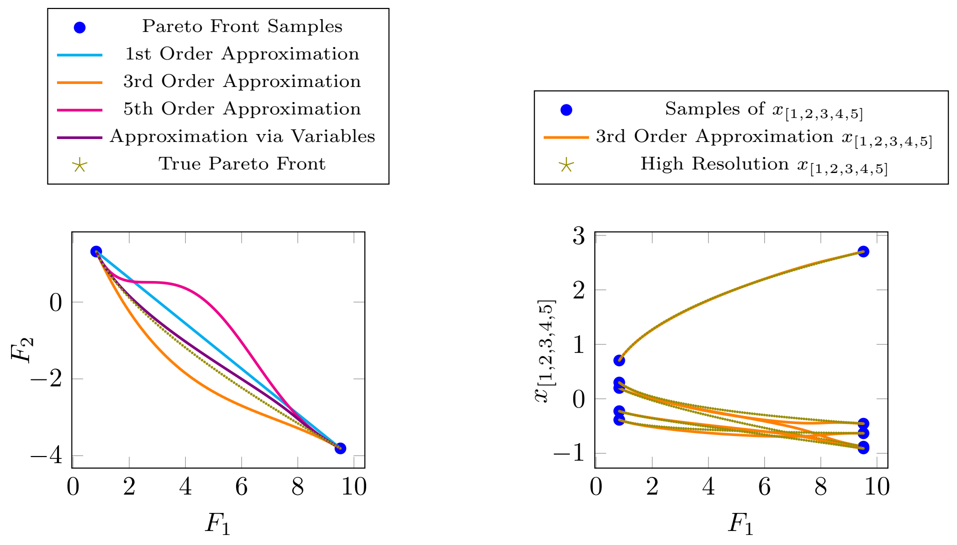
Figure 3.5a shows the graphs of all possible direct Pareto front approximations for Example 3.1. The cyan line displaying a polynomial of first grade was created by using only the values of both samples $x^{[1]}$ and $x^{[2]}$. This approach represents the method usually applied to connect two samples. Even when not displayed as a line, this would be the only possible connection when no sensitivity analysis information is available. In this example, it is clear that to create a correct figure of the Pareto front and assume equidistant samples, at least two more samples are necessary. Thus the computational effort would be doubled.

However, the graph of the 3rd-order approximation displayed in orange indicates the correct overall shape of the Pareto front. This fact shows how much the 1st-order parametric sensitivity information helps interpret the Pareto front and its tradeoffs. In contrast to the local approximation in Figure 3.1, it becomes easier to interpret the information because the graph resulting from the combination of both samples is directly visible. The fact that the approximation via

variables does not give any further information in this example is also supported by the fact that the graph of the approximation via variables displayed in violet is almost identical to the orange one, which displays the approximation via derivatives of F. Figure 3.5b displays the approximated variables used. Since the linear developing variable is approximated exactly and the atan-shaped with the same accuracy as the approximation of third order for the Pareto front in Figure 3.5a, it is clear that, in this example, there will be almost no difference seen between the direct approximation of the Pareto front and the approximation done via variables. The difference that is visible is probably caused by numerical noise in the computation.
Increasing the order of approximation to 5 by taking the second derivative into account improves the approximation further. This is displayed by the magenta-coloured graph in Figure 3.5a. It approximates the curves with a higher accuracy leading to a better approximation of the overall shape. However, this example does not provide an improved insight into the general shape of the Pareto front. Of course, this would not be expected since the 3rd order approximation already displays the general shape correctly.

All in all, this example illustrates the potential of using parametric sensitivity information for approximating the Pareto front. Even if a human might already guess the general shape of the Pareto front by looking at the local approximations in Figure 3.1, the smooth connection between samples, as performed here, is advantageous when the shape is processed by another algorithm, a non-human decision-maker or consists of more complex shapes.

However, since polynomials are used for the approximations, the usual pitfalls of polynomial interpolation (see [92]) need to be considered here. These can be very well illustrated by applying the proposed approach to the second Example 3.2.



(a) Global Pareto front approximations of 1st-, 3rd- and 5th-order.

(b) Global variable approximations of 3rd-order.

Figure 3.6: Global approximations of different orders orders for Example 3.2.

The results shown in Figure 3.6a for Example 3.2 show a very different behaviour than the graphs in Figure 3.5a. Here, the approximations of a higher order, which are computed directly with the derivative information of F, approximate the Pareto front inaccurately and show beginning oscillating behaviour. The orange graph, which displays the approximation of 3rd-order, is overall a not too bad approximation as the distance to the true Pareto front is in a similar range as the distance of the linear approximation in cyan. However, when looking at the details, it is not a convex function, whereas the Pareto front is convex shaped. Thus the general shape of the Pareto front is not approximated correctly. However, the information of convexity is not provided by the linear approximation. Thus it is hard to say which one is the better approximation here. However, compared to the information obtained when looking at the local approximations in Figure 3.2, the connection of the approximations between the two samples does not improve the overall figure. Rather the opposite, as the local approximations clearly show the local convexity of the Pareto front.
Even worse is the approximation of 5th-order shown as the magenta-coloured line. Here the locality of the parametric sensitivity information is visible. Thus, the graph is quite accurate in a small area around the samples and the best approximation within this area. However, the overall approximation does not correlate to the shape of the Pareto front at all. Since the areas where this approximation is accurate are very small, the connected approximation does not provide more information than the local approximations shown in Figure 3.2. The local approximations also give a hint at why the polynomial interpolation might not work here. The high curvature in the left sample already leads to a somewhat extreme local approximation that influences the overall shape. As a result, the transition into the almost linear part of the Pareto front is not correct.

Besides an example for the limitations of polynomial interpolation, the results in Figure 3.6a show the fact that the approximation via the variables does not suffer from these problems. On the contrary, they provide the best approximation of the Pareto front if only the distance to the true Pareto front is taken into account. The reason for this can be seen in Figure 3.6b where the global approximations of the variables are shown in orange. The approximation here shows better accuracy than the 3rd-order approximation of the Pareto front. Thus by evaluating F for these approximated variables, higher precision can be obtained. However, similarly to the 3rd-order approximation, the graph is not convex and might give a wrong idea about the overall shape, even when the true Pareto front is approximated with reasonable accuracy. However, the example shows that the approximation via variables differs fundamentally from the approximation via derivatives of F and, in some cases, can provide a better approximation to the Pareto front. The question of which approach results in the better approximation will strongly depend on the example, and especially on how much nonlinearity is introduced by the objective function and how the variables develop over τ will influence the result of the approximations.

Interestingly, when inserting one more sample in the middle, the problems of the approximation via variables vanish as shown in Figure 3.7, but the 5th-order approximation still shows oscillating behaviour, and the 3rd-order approximation is still not a perfectly convex graph even if the difference is hard to see.

Another interesting idea is illustrated by the graphs in Figure 3.7. When the position of the third sample between the other two is clear, there arises a question: If exactly one more sample could be included, where should it be included? When looking at the approximations in Figure 3.7, the answer is that it should be in the left interval. Of course, in reality, the true Pareto front would not be known, but the fact that all four approximations are almost identical gives a

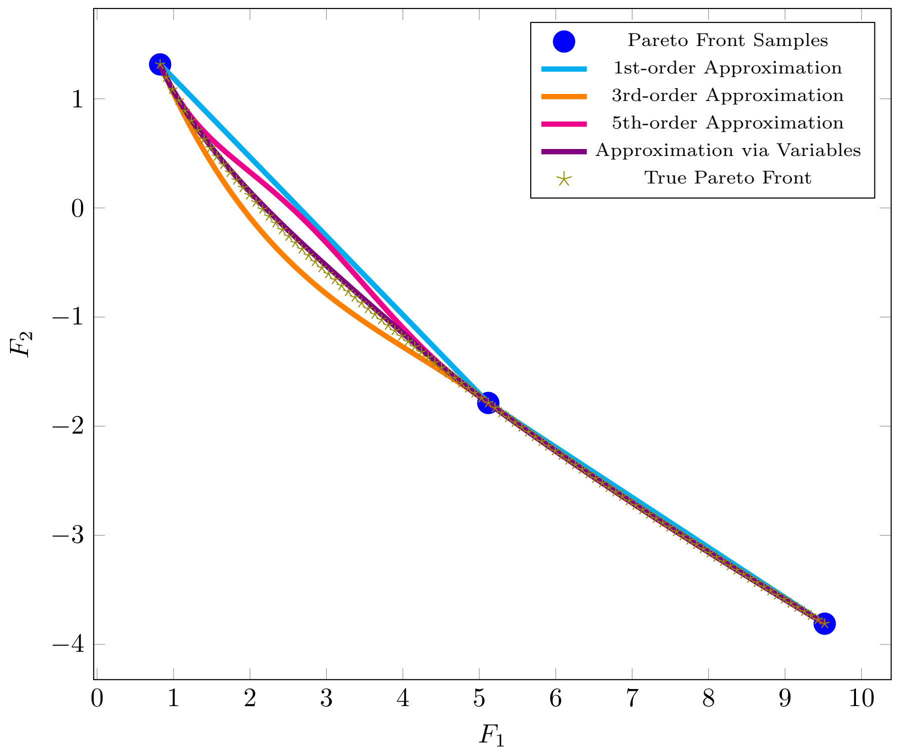


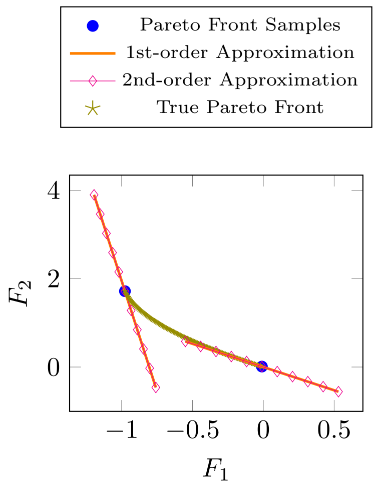
Figure 3.7: Global Pareto front approximations of 1st-, 3rd- and 5th-order for Example 3.2 based on three samples, compared to the true Pareto front.

clear hint that they are correct. This fact leads to the question: Can the differences between the different approximations be used to steer an adaptive algorithm? This algorithm would decide automatically where the next sample should be included, thus using the available computational power more effectively than an equidistant sampling would do. Such an approach is even more plausible when looking at the shape of the Pareto front of Example 3.2. Here the right side of the Pareto front is almost linear and can thus be approximated accurately with only two samples. Any more samples in this area would be a waste of computational power. In contrast, the left side has a higher curvature and samples in this area would provide more information.
However, such an approach is not trivial since the difference between multiple approximations hints at the fact that at least one of them is not correct, but that the other direction, that identical approximations prove the correctness is not accurate as was discussed in [5]. However, the information of all approximations could be gathered and used by an approach using neural networks or other machine learning techniques as was done for interior point algorithms in [79].

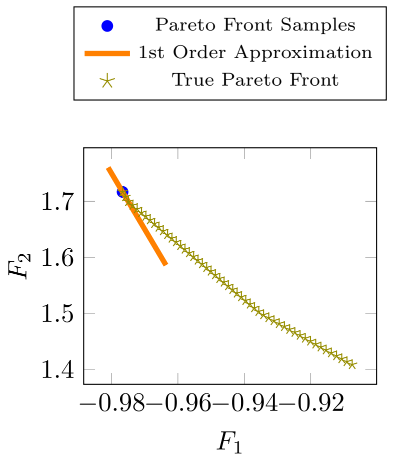
This section showed the potential of using local information from parametric sensitivity analysis to approximate the Pareto front globally. It was shown that this approximation reduces the number of samples needed for an accurate approximation. It also showed some problems when using this local information on a global scale. However, these problems are the same as with every interpolation technique since the values of the samples of the Pareto front are also only local information. All in all, the proposed methods represent another tool when aiming for the reduction of computational effort in multi-objective optimisation.

3.3 Limitations of Approximations using Parametric Sensitivity Analysis

The previous sections have shown the potential in using parametric sensitivity analysis information to approximate the Pareto front. This section discusses an example where the proposed methods hit some limits, especially in the case that some conditions on the smoothness of some functions involved are not fulfilled correctly.



(a) Local Pareto front approximations of 1st- and 2nd-order.

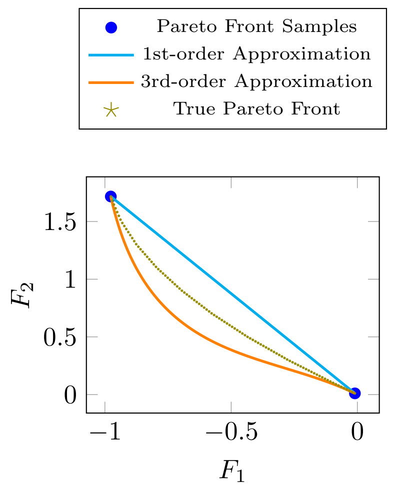


(b) Local Pareto front approximation of 1st-order at left sample.

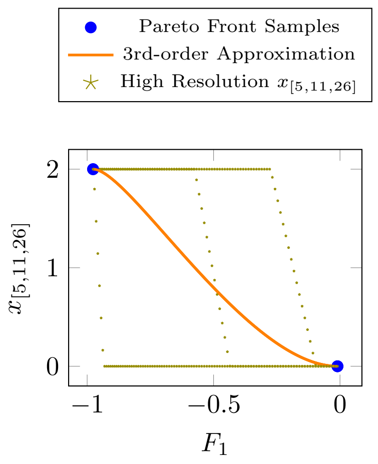
Figure 3.8: Local Pareto front approximations of 1st- and 2nd-order for Example 3.3.

In Figure 3.8a the local approximations for the Rocket Car Example 3.3 are displayed. At first glance, the overall result looks similar to the one from the NBI Example 3.2 in Figure 3.2. This similarity is that the linear approximation of the Pareto front in orange looks accurate. However, the 2nd-order approximation in magenta and the 1st-order approximation in orange have the same graph for both samples. This behaviour is unexpected since the Pareto front is not linear, and at least at the left sample, a higher curvature should be computed. A closer investigation reveals that in fact $\frac{\partial \lambda}{\partial a} < 10^{-5}$ which is close to a numerical 0 for the used settings. A closer look at the Pareto front reveals the origin of this behaviour. To get this closer look, the true Pareto front was sampled with a higher resolution of 1000 samples. Figure 3.8b shows the first 100 samples from the upper left sample used for local approximation. This close view shows that the Pareto front is indeed piecewise linear and not curved. What is displayed shows that the linear approximation is exact for the first samples. Afterwards, two more linear intervals are displayed. Thus, it is clear that a 2nd-order approximation will be the same as a 1st-order approximation because, locally, there is no curvature. It also emphasises the locality of the parametric sensitivity information.

Nevertheless, even if the problem described above renders the 2nd-order information useless for usage in a better approximation, the global approximation of 3rd-order might still be accurate for the example. The results for this are shown in Figure 3.9a. As the 1st-order approximation,



(a) Global Pareto front approximations of 1st- and 3rd-order.



(b) Global variable approximations of 3rd-order.

Figure 3.9: Global Pareto front approximations of different orders for Example 3.3.

the result for the global approximation is similar to the one from the NBI example in Figure 3.6a. The approximation of 3rd-order is approximately as good as the approximation of 1st-order when evaluating the distance to the true Pareto front. In addition, the fact that information by the second derivatives does not provide a better approximation is still useful since it reveals insights about the structure of the Pareto front.

In terms of problems, the global approximation of the variables is more interesting. Figure 3.9b shows the global approximation of x_5, x_{11} and x_{25}. These three variables represent the control value $u(t)$ at different discrete time points. When looking at the graphs of the true Pareto front, it is clear that the implicit function $x(\tau)$ is not continuously differentiable in all values of τ. Here the discontinuity in the derivative of x is also larger than the one of the Pareto front in objective space, shown in Figure 3.8b. When looking at the approximations displayed as orange graphs, two problems become visible. The first is that all three approximations are the same. The second is that the approximations are not accurate. Both originate in the fact that the graphs for all three variables are piecewise linear and only differ in the position they switch from constant to a linear change and back to constant. Since this information is global and can not be captured with the local information of parametric sensitivity analysis, the result is as displayed.

The discussed example shows some limitations of the proposed method. It emphasises the locality of the parametric sensitivity analysis information. Also, it reveals problems that arise when the implicit maps of the manifolds defined by the Pareto front are not continuously differentiable. However, it also shows that global approximation might still be useful when the discontinuities are not significant. When it comes to applying the algorithms, and all values are discrete, the border between a real discontinuity and one introduced by discretisation is not a sharp one. Still, if no continuous differentiability is given, classical methods should be preferred. It should also be noted that the shown example was purposefully picked. It shows the behaviour resulting from a so-called “bang-bang” control, a well-known special case in optimal control theory. Thus, the discussed behaviour is not an purely academic example but will likely show up in real applications (see for example [8]).

Note 3.7
When interpolating the variables of a given problem, the feasibility of the obtained solution is not guaranteed. In [7] parametric sensitivities are once again used to overcome this problem. Based on the additional idea, a feasibility refinement as presented in [18] is performed after the interpolation. In the example used in [7], this was possible for all points between two samples. However, this chapter emphasises that parametric sensitivity information is local information and should thus be used with care, especially when doing something like feasibility refinement. One idea is to use the refinement technique in Pareto-navigation (see [1]) combined with the interpolation, thus speeding up the process. However, this specialised usage would require much more specialised implementation and is thus not covered in this thesis.

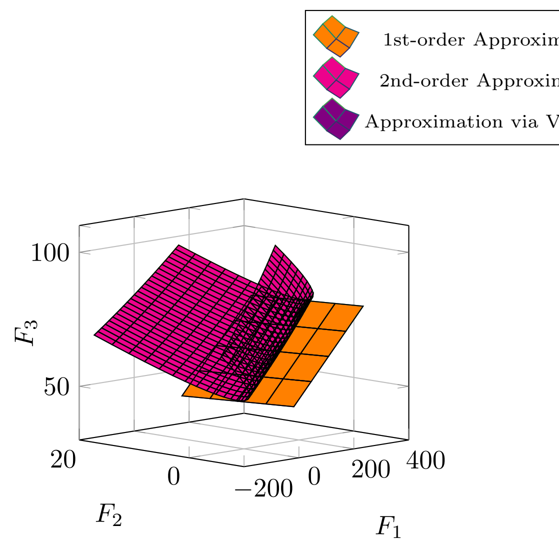
3.4 Ideas to Extend the Approaches to more than two Dimensions

In Sections 3.1 and 3.2 the approximations were limited to bi-objective examples. This section shows how the local approximation can be generalised for higher dimensions and gives some ideas on how to do the same for the global approximation approach.

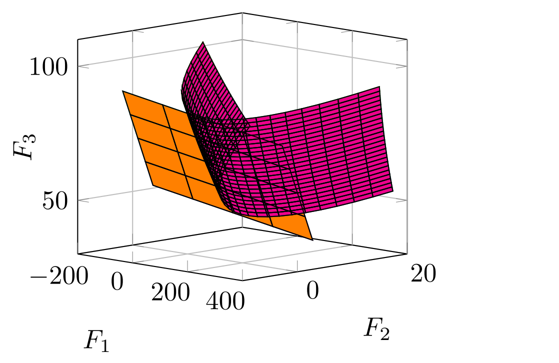
3.4.1 Local Pareto Front Approximation in Higher Dimensions

So far, only the bi-objective case was considered for the local approximation. However, it can be generalised by extending the one-dimensional Taylor series development to a multidimensional development.
Using the definition from the beginning of Section 3.1.1, the Pareto front was described as a function $\mathcal{M} : \mathbb{R}^{N_F-1} \to \mathbb{R}^{N_F}$. It becomes clear that for a multidimensional Taylor series development in $a(\tau)$ the partial derivatives for all a_i $i = 1, \ldots, N_F - 1$ are needed. However, the derivatives $\frac{\partial F}{\partial a}$ as described in [33] were never limited to the bi-objective case. Combined with the fact that a can be computed for every value of τ it becomes clear that computing the derivative $\frac{\mathrm{d}F}{\mathrm{d}a}$ and thus also $\frac{\partial F}{\partial a_i}$ $i = 1, \ldots, N_F - 1$ is also not feasible in higher dimensions. Since parametric sensitivity analysis yields also $\frac{\partial \lambda}{\partial a_i}$ for all $i = 1, \ldots, N_F$, the second order approximations are also no problem for the local approximation.
The approximation of the variables is still possible as well. The Taylor-series development needs only the same small extension to multiple dimensions.

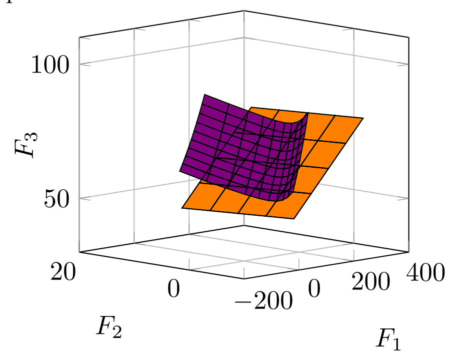
To show the applicability to higher dimensions, Figure 3.10 displays the local approximation for Example P.38 in [114] with $N_F = 3$. The 1st-order approximation is shown in orange, the 2nd order approximation in magenta and the approximation by variables in violet. As in the examples for $N_F = 2$, the linear approximation displays the tradeoffs between the individual objective functions, providing valuable information to the decision-maker.
The quadratic approximation in Figure 3.10 shows clearly that the curvature is very different for the individual directions. This information is precious in higher dimensions since it shows



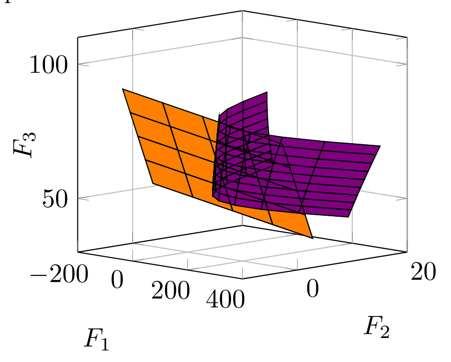
(a) Approximation of 1st- and 2nd-order and approximation via variables.



(b) Approximation of 1st- and 2nd-order and approximation via variables.



(c) Approximation of 1st-order and approximation via variables.



(d) Approximation of 1st-order and approximation via variables.

Figure 3.10: 3D Local approximation of the Pareto front of Example 3.4 viewed from two different angles.

in which direction new points might add more value. It can also be used in Pareto front navigation [1] since it indicates in which direction the tradeoffs change the most and how they change.

A bit different is the display of the approximation by variables in violet. It needs to be mentioned here that the approximation with variables was cut at the edges since otherwise the variables would have violated the box constraints. This would have let to the evaluation of $\ln(x)$ for $x \leq 0$. Of course, it is also a difference of the approximations, that the box bounds can and must be considered when approximating the variables. At first, this seems like a constraint. However, it also gives the advantage that the decision-maker sees this information if the approximation happens near the feasible set's border.

3.4.2 Global Pareto Front Approximation in Higher Dimensions

Section 3.4.1 describes that the local approximation of Pareto fronts can easily be extended to dimension $N_F > 2$. However, for the global approximation between samples, the increase of N_F introduces a few problems. This section discusses these problems and briefly gives ideas on how they can be solved. However, the ideas discussed are not implemented or investigated further within this thesis.

The first problem for interpolation between neighbouring samples in higher dimensions is the notion of "neighbour" itself, because the term is not as well defined as in the bi-objective case where samples are neighbours to each other. One way to overcome this problem is using triangulation as described by Nowak and Küfer [96] resulting in $N_F - 1$ dimensional simplices where the corners are samples of the Pareto front, and the edges can be utilised to define neighbours. In [96] the simplices are computed after the Pareto front samples are projected onto a $N_F - 1$ dimensional hyperplane. For methods like the Pascoletti-Serafini (PS(a, r)) approach, which already are defined by the variation of the hyperparameter a along a hyperplane, these parameters can also be used to triangulate the solutions. This could be easier to compute since the parameter can be generated more regularly, which makes an elaborate triangulation algorithm unnecessary and thus leads to a possibly better computation time. However, the triangulation might turn out to be the bottleneck within the computation of Pareto fronts.

The simplices computed by a triangulation algorithm do not only define neighbours but also solve the problem of defining a region in which a global approximation is defined, the equivalent to intervals in the bi-objective case. However, trying to compute a polynomial approximation for a whole simplex leads to another problem. That is, multidimensional nonlinear polynomials introduce a high number of parameters that grow faster than the number of information points provided through samples and their parametric sensitivities. Thus it is not possible to define the polynomials for the whole simplex.
A possible workaround could be to define the polynomials only along the edges of the simplex and describe the Pareto front based on these edges. This workaround would lead to a similar approach as described by Peitz [99] with the difference that the edges are approximated and not sampled. For the case that $N_F > 3$, in which the edges are again sets of dimensions higher than 2, this approach would need to be applied recursively.
Besides the possibility to see the overall structure of the Pareto front based on the edges alone, it is also possible to interpolate linearly between the approximated functions, thus getting a function defined on the whole simplex.
Note that for the described approach, it might be beneficial to use the triangulation based on the hyperparameter a since that would lead naturally to an interpolation along the base vectors of the hyperplane, which are also the directions in which the derivatives are computed, thus ensuring that all the computed directional derivatives are used.

The globalised approach for an approximation is not as naturally extendable to higher dimensions as the local approximation. The proposed ideas need further investigation to clarify how the global approximation can be generalised to higher dimensions.

3.5 Conclusion and Outlook on Using Parametric Sensitivity Analysis Information for better Pareto Front Approximations

This chapter shows how the information obtained by a parametric sensitivity analysis of samples obtained via the Pascoletti-Serafini scalarisation can be used to improve the approximation and understanding of Pareto fronts.

First, it is shown how a local approximation up to the second degree can be computed from the sensitivity analysis and how this enhances the information given to the decision-maker. The chapter also sets out that the sensitivities of the individual objective functions can be used and shows how the approximation of the variables can be used to obtain another approximation of the Pareto front. The different approximations are compared in two examples.

Second, this chapter extends the idea of a local approximation to computing polynomial approximations for the whole region between two samples of a Pareto front. It is shown how this can reduce the number of samples needed to get a good approximation. However, the limitations and pitfalls of this method are also investigated.

Last, it is shown that the local approximation also works in dimensions $N_F > 2$. For the global approximation, ideas to extend the approach to higher dimensions are discussed.

All in all, this chapter provides various new tools for the decision-maker to guide the decision and find the preferred Pareto-point. Furthermore, these tools do not require high extra computational effort due to the very efficient way of computing parametric sensitivities within SQP-based solvers.

The most natural extension of the work presented in this thesis is generalising the described methods to higher dimensions. Possible ideas for this have already been discussed in Section 3.4.1. For a complete insight, it would also be beneficial to investigate the connection of the reduction of needed samples and the Nyquist criteria, or in general to investigate the frequencies of the Pareto front.

This thesis uses polynomials to compute an approximation between samples of the Pareto front. However, polynomials are not the only approximations that use derivative information. Bézier curves (see [35]), for example, can use the derivative information to compute the control points which influence the curve. In contrast to polynomials, Bézier curves have several good convergence properties, emphasised by their widespread use in computer graphics. One property is that it is guaranteed that the curve lies within the convex region defined by all control points. Thus, they allow an error-estimation and possible convergence criteria when applied to Pareto front approximation. However, Bézier curves are more complicated than polynomials, leading to problems, especially when extending the methods to higher dimensions.

Besides altering the approach itself, there are also possibilities in using the output of the proposed method. For example, it could be possible to define new adaptive scalarisation algorithms similar to the one described by Eichfelder [33] but using 2nd-order derivatives. It could also be possible to use a machine-learning-based approach that utilises the parametric sensitivity analysis information. Especially the differences between the approximations directly in the objective space or via the variables might give useful information at this point.

Another idea based on the parametric sensitivities used in this chapter is to define a new algorithm for very specific bi-level optimisation problems. By providing the sensitivities of the hyperparameter a to the tradeoffs of the individual objectives, it might become possible to define a gradient-descent like an algorithm to optimise an objective function that depends on the tradeoffs. For example, in a bi-objective setting, the single tradeoff between both objectives could be minimised. This application of sensitivity information would open up a whole new set of possibilities for the decision-maker. Especially in settings where the decision-maker is an algorithm itself, and only a single Pareto front sample is needed, like in Model Predictive Control algorithms with several objectives to minimise, this could be a beneficial tool to find Pareto points with specific criteria.

Chapter 4

Advances in Multi-Objective Sequential Quadratic Programming

Contents

In Section 2.5 all algorithms depend on parameters that are given before the main optimisation algorithm is executed. The choice of these parameters is crucial for the obtained points, their distribution and spread over the Pareto front and for whether or not the entire Pareto front is covered. This a-priori choice of parameters is, in some cases, challenging to obtain, so that complex calculations must be carried out in advance (see Section 2.5.3).

Although most of these problems can be circumvented by using heuristic methods such as genetic algorithms [27, 112, 126], these methods lack a convergence proof and have a low rate of convergence [14, 46, 114]. They also struggle when the problems considered consist of a high number of constraints, as is the case in discretised optimal control problems [90].
The Multi-Objective Sequential Quadratic Programming Algorithm (MOSQP) proposed by Fliege and Vaz [46] aims at tackling all these problems by combining the efficiency of scalarisation approaches with the subsequent solution by NLP-Solvers and the generality of genetic algorithms [112]. In simple terms, this is done by iteratively improving a set of nondominated points by repeatedly applying one SQP step in the direction of every individual objective and later in the “direction” of local Pareto-optimality. In between these iterations, a cleanup strategy similar to those in genetic algorithms is performed. Additionally, Fliege and Vaz [46] prove the

convergence of the proposed algorithm.

Already a performant algorithm, the original MOSQP algorithm still faces some problems addressed by this thesis, namely the proposed cleanup strategy which might delete promising points in early iterations. In addition, the proposed initialisation strategies can be extended for more generality. The former is addressed in Section 4.2 which proposes to use the cleanup strategy of the NSGA-II [27] which is based on nondominated sorting rather than solely on dominance. Section 4.3 explains how the algorithm is implemented using the NLP-Solver WORHP (see 2.3). The complete algorithm is given in Section 4.4 before the existing initialisation strategies are extended in Section 4.5. The impact of all proposed advancements is discussed in Section 4.7.

Besides enhancements in the theoretical setup of the algorithm, this thesis shows how the algorithm is implemented using the solver WORHP (see Section 2.3 and [19]) for building the necessary QPs and for exploitation of features such as step size and regularisation of corresponding derivative matrices (see Section 2.3). It also shows how the algorithm can benefit from parallelisation on modern computers, which is only mentioned in passing in the original paper.

This chapter first provides fundamental definitions, then introduces the original ideas of the MOSQP algorithm and shows where and how they can be improved. It is strongly based on the paper of Fliege and Vaz [46] but extends it by the ideas mentioned above.

4.1 Fundamentals of the MOSQP Algorithm

Chapter 2 introduces some important fundamentals for continuous nonlinear constrained optimisation problems. However, this section adds some basic definitions which are adapted for multi-objective optimisation.

4.1.1 Handling Constraints

One main problem in constrained multi-objective optimisation is the handling of constraints. More precisely, the question of how to compare infeasible and feasible points during the iteration process is of special interest. For the MOSQP Algorithm, Fliege and Vaz [46] use the following ideas and definitions.

In an abstract way, each inequality and equality constraint can be interpreted as an additional objective to optimise. But, while objective functions are to be minimised, constraint functions usually must reach the value 0 (equality constraints) or be lower than 0 (inequality constraints). In order to deal with the latter

$$c^+(x) := \max\{0, c(x)\}, \tag{4.1}$$

is defined for any real valued function $c \in \mathbb{R}$. Thus, every feasible point of (MONLP) satisfies

$$g_i^+(x) = 0, \quad i = 1, \ldots, N_g, \quad \text{and} \quad |h_j(x)| = 0, \quad j = 1, \ldots, N_h. \tag{4.2}$$

With these definitions, the concept of weak Pareto dominance (see definition 2.35) can be extended to

$$\bar{F}(x) \preceq_{\bar{F}} \bar{F}(y) \quad \Leftrightarrow \quad \bar{F}_k(y) - \bar{F}_k(x) \leq 0, \quad k = 1, \ldots, N_F + N_g + N_h, \tag{4.3}$$

with

$$\bar{F}(x)^\top := \left(F(x)^\top, \Phi(x)^\top\right) \text{ and } \Phi(x)^\top := \left(g^+(x)^\top, |h(x)^\top|\right). \tag{4.4}$$

Similarly, the concept of strong dominance can be defined.
It can be seen directly that for points that are feasible for (MONLP) these definitions are equivalent to the usual concept of dominance. For infeasible points, the concept is extended in the way that an infeasible point never weakly dominates a feasible point while a feasible point can weakly dominate an infeasible one.

4.1.2 Measuring the Progress over Iterations

In order to measure the progress over a sequence of iterates, the following variant of the classical ℓ_1 merit function is used:

$$\phi(x,\sigma) = \sum_{l=1}^{N_F} F_l(x) + \sigma \left(\sum_{i=1}^{N_g} (g_i(x))^+ + \sum_{j=1}^{N_h} |h_j(x)| \right), \tag{4.5}$$

where σ is a positive penalty parameter (see [51]). In addition, in a single-objective case the function

$$\phi_l(x,\sigma) = F_l(x) + \sigma \left(\sum_{i=1}^{N_g} (g_i(x))^+ + \sum_{j=1}^{N_h} |h_j(x)| \right), \tag{4.6}$$

is used for minimising the lth objective with $l \in \{1, \dots, N_F\}$.

Note 4.1
The functions ϕ and ϕ_l are usually non-differentiable. However, the directional derivatives at x in any direction $0 \neq d \in \mathbb{R}^{N_F}$ exist (see [10] for details) and are in the remainder of this chapter denoted as

$$\frac{\partial \phi}{\partial d}(x,\sigma). \tag{4.7}$$

4.1.3 Improving a Set of Nondominated Points

Within the MOSQP algorithm, the idea is to use SQP like techniques to first spread points along the Pareto front and afterwards drive them toward local Pareto-optimality. This section first shows under which conditions a search direction is suitable to improve a given point along the Pareto front. It is then shown how such a direction can be computed.

Theorem 4.2
Let $X \subset \mathbb{R}^{N_x}$ be a set of nondominated points. If $d \in R^{Nf}$ satisfies for some $x^{[k]} \in X$ that is feasible or infeasible for (MONLP) and some $l \in \{1, \dots, N_F\}$

$$\nabla_x F_l(x^{[k]})^\top d < 0, \tag{4.8}$$

$$g_i(x^{[k]}) + \nabla_x g_i(x^{[k]})^\top d \leq 0, \quad i = 1, \dots, N_g, \tag{4.9}$$

$$h_j(x^{[k]}) + \nabla_x h_j(x^{[k]})^\top d = 0, \quad j = 1, \dots, N_h, \tag{4.10}$$

and $\exists \bar{t} > 0$ such that

$$\phi_l(x^{[k]} + td, \sigma) \leq \phi_l(x^{[k]}, \sigma) + \mu t \frac{\partial \phi_l}{\partial d}(x, \sigma), \quad \forall t \in (0, \bar{t}], \tag{4.11}$$

with some $\mu \in (0, 1)$ and some $\sigma > 0$.

Then d is called a **descent direction** for F_l and there exists a $t_0 > 0$ such that

$$\bar{F}(x^{[k]}) \not\leq_{\bar{F}} \bar{F}(x^{[k]} + td), \quad \forall t \in (0, t_0]. \tag{4.12}$$

In addition with $a : \mathbb{R}^{N_g} \to \{0,1\}^{N_g}$ and $b : \mathbb{R}^{N_h} \to \{0,1\}^{N_h}$ defined by

$$a_i(x) := \begin{cases} 1, & g_i(x) = 0, \\ 0, & g_i(x) \neq 0, \end{cases} \qquad b_j(x) := \begin{cases} 1, & h_j(x) = 0, \\ 0, & h_j(x) \neq 0, \end{cases} \tag{4.13}$$

with $i = 1, \ldots, N_g$ and $j = 1, \ldots, N_h$. Define also $r(x) := (a(x)^\top, b(x)^\top)^\top$. Then for all $\epsilon > 0$ there exists a $t_0 > 0$ such that we have

$$\Big(\Phi(x^{[k]}) - \Phi(x^{[k]} + td) + \epsilon r\Big)_\ell > 0,$$

for all $\ell = 1, \ldots, N_g + N_h$ and $t \in (0, t_0]$.

Proof: See [46].

Theorem 4.2 shows how a search direction d must look like in order to find a new point $x^{[k+1]}$, which represents progress for at least one objective and is not dominated by the previous point $x^{[k]}$.

The main idea of the MOSQP algorithm is to compute these search directions by using the same quadratic approximations to the objective functions and linear approximations to equality and inequality constraints as those used in a SQP method (see Chapter 2.1.2).
In concrete terms this means that a search direction as in 4.2 with respect to the l-th objective function ($l \in \{1, \ldots, N_F\}$) can be computed by solving the quadratic optimisation problem:

$$\begin{array}{lll} \min\limits_{d \in \mathbb{R}^{N_x}} & \nabla_x F_l(x^{[k]})^\top d + \frac{1}{2} d^\top H_l d, & \\ \text{subject to} & g_i(x^{[k]}) + \nabla_x g_i(x^{[k]})^\top d \leq 0, & i = 1, ..., N_g, \\ & h_j(x^{[k]}) + \nabla_x h_j(x^{[k]})^\top d = 0, & j = 1, ..., N_h, \end{array} \tag{4.14}$$

where H_l is the Hessian of the Lagrange function (2.4).
In order to connect the solution of (4.14) to Theorem 4.2, Fliege and Vaz [46] state the following Lemma.

Lemma 4.3
For each $l = 1, \ldots, N_F$, let d^* be an optimal solution to problem (4.14). Then d^* satisfies the following conditions:

1. If $x^{[k]}$ is feasible for problem (MONLP), then $\nabla_x F_l(x^{[k]})^\top d^* < 0$.

2. Let $x^{[k]}$ be infeasible for problem (MONLP), and let r be defined as in Theorem 4.2. Then for all $\epsilon > 0$ there exists a $\bar{t} > 0$ such that $\Phi(x^{[k]}) - \Phi(x^{[k]} + td^*) + \epsilon r > 0$, $\forall t \in (0, \bar{t}]$.

3. Let $d^* \neq 0_{N_x}$, let r be defined as in Theorem 4.2, and define $\hat{r} : \mathbb{R}^{N_x} \to \mathbb{R}^{N_F+N_g+N_h}$ by $\hat{r}(x) := (0_{N_F}^\top, r(x)^\top)^\top$, with $0_{N_F} = (0, \dots, 0)^\top \in \mathbb{R}^{N_F}$. Then for all $\epsilon > 0$ there exists a $\bar{t}$ such that
$$\bar{F}(x^{[k]}) \not\prec_{\bar{F}} \bar{F}(x^{[k]} + td^*) - \epsilon\hat{r}, \quad \forall t \in (0, \bar{t}]. \tag{4.15}$$

4. If $d^* = 0_{N_x}$, then $x^{[k]}$ is feasible for (MONLP) and there does not exist a feasible d for (4.14) such that $\nabla_x F_l(x^{[k]})^\top d < 0$.

5. If $d^* \neq 0_{N_x}$, then there exists a $\bar{t}$ such that for all $t \in (0, \bar{t}]$ and $\sigma > 0$ sufficiently large, we have
$$\phi_l(x^{[k]} + td, \sigma) \leq \phi_l(x^{[k]}, \sigma) + t\mu\nabla\phi_l(x^{[k]}, \sigma)d, \tag{4.16}$$
for all $\mu \in (0, 1]$.

Proof: See [46].

Note 4.4
Problem (4.14) is the resulting QP from an intermediate iteration of an SQP method (see Section 2.1.2) for problem
$$\begin{array}{lll} \min\limits_{x\in\mathbb{R}^{N_x}} & F_l(x), & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, ..., N_g, \\ & h_j(x) = 0, & j = 1, ..., N_h. \end{array} \tag{4.17}$$

4.1.4 Driving a Nondominated Set of Points to local Pareto Optimality

After showing how a set of points can be improved with respect to the spread, this section focuses on how points can be driven towards local Pareto-optimality. Analogous to the previous section, this section first introduces conditions under which a descent direction improves a given point towards Pareto-optimality. Then it is shown how such a descent direction can be calculated using SQP-like techniques.

Theorem 4.5
Given $x^{[k]} \in X \subset \mathbb{R}^{N_x}$ and a descent direction $d \in \mathbb{R}^{N_x}$ which satisfies the following conditions. There exists an $l_0 \in \{1, \dots, N_F\}$ such that
$$\nabla_x F_{l_0}(x^{[k]})^\top d < 0, \tag{4.18}$$
$$\nabla_x F_l(x^{[k]})^\top d \leq 0, \quad l = 1, \dots, N_F,\ l \neq l_0, \tag{4.19}$$
$$g_i(x^{[k]}) + \nabla_x g_i(x^{[k]})^\top d \leq 0, \quad i = 1, \dots, N_g, \tag{4.20}$$
$$h_j(x^{[k]}) + \nabla_x h_j(x^{[k]})^\top d \leq 0, \quad j = 1, \dots, N_h, \tag{4.21}$$
and $\exists\bar{t} > 0$ such that
$$\phi(x + td, \sigma) \leq \phi(x, \sigma) + \mu t\nabla\phi(x, \sigma)d, \quad \forall(t \in 0, \bar{t}], \tag{4.22}$$
with some $\mu \in (0, 1)$ and some $\sigma > 0$.
In addition define $\omega : \mathbb{R}^{N_F} \to \{0, 1\}^{N_F}$ by
$$\omega_l(x) := \left\{ \begin{array}{ll} 1, & \nabla_x F_l(x)^\top d = 0, \\ 0, & \nabla_x F_l(x)^\top d \neq 0, \end{array} \right. \quad l = 1, \dots, N_F, \tag{4.23}$$

and $\bar{r} : \mathbb{R}^{N_x} \to \{0,1\}^{N_F+N_g+N_h}$ by $\bar{r}(x) \coloneqq (\omega^\top, a^\top, b^\top)^\top$, with a, b defined as in Theorem 4.2. Then for all $\epsilon > 0$ there exists a $t_0 > 0$ such that

$$\bar{F}(x^{[k]} + td) \preceq_{\bar{F}} \bar{F}(x^{[k]}) + \epsilon\bar{r}, \quad \forall t \in (0, t_0]. \tag{4.24}$$

Proof: See [46].

Similar to the search direction for Theorem 4.2, the question is now how to find a search direction d suitable for the conditions in Theorem 4.5.
For this consider the QP

$$\begin{array}{lll} \min\limits_{d \in \mathbb{R}^{N_x}} & \sum_{l=1}^{N_F} \nabla_x F_l(x^{[k]})^\top d + \frac{1}{2} d^\top H_l d, & \\ \text{subject to} & g_i(x^{[k]}) + \nabla_x g_i(x^{[k]})^\top d \le 0, & i = 1, ..., N_g, \\ & h_j(x^{[k]}) + \nabla_x h_j(x^{[k]})^\top d = 0, & j = 1, ..., N_h, \end{array} \tag{4.25}$$

where H_l are symmetric positive definite matrices.

Analogous to Problem (4.14), Problem (4.25) is connected to the conditions in Theorem 4.5 by the following Lemma from [46].

Lemma 4.6
Let d^* be a solution to problem (4.25). Define $\vartheta_l(d) \coloneqq \nabla_x F_l(x^{[k]})^\top d + (1/2) d^\top H_l d$, $\vartheta_l^* \equiv \vartheta_l(d^*)$, $l = 1, \dots, N_F$ and $\vartheta^* = \sum_{l=1}^{N_F} \vartheta_l^*$. Then the following assertions hold:

1. If $x^{[k]}$ is feasible to problem (MONLP), then $\vartheta^* \le 0$.

2. Let $x^{[k]}$ be infeasible for problem (MONLP), and define r as in Theorem 4.2. Then for every $\epsilon > 0$ there exists a $\bar{t} > 0$ such that

$$\Phi(x^{[k]}) - \Phi(x^{[k]} + td^*) + \epsilon r > 0, \tag{4.26}$$

   for all $t \in (0, \bar{t}]$.

3. Let $d^* \ne 0_{N_x}$, let r be defined as in Theorem 4.2, and define

$$\hat{r} : \mathbb{R}^{N_x} \to \mathbb{R}^{N_F+N_g+N_h}, \text{ by } \hat{r}(x) \coloneqq (0_{N_F}^\top, r(x)^\top)^\top, \tag{4.27}$$

   with $0_{N_F} = (0, \dots, 0)^\top \in \mathbb{R}^{N_F}$. Then for all $\epsilon > 0$ there exists a $\bar{t}$ such that

$$\bar{F}(x^{[k]}) \not\preceq_{\bar{F}} \bar{F}(x^{[k]} + td^*) - \epsilon\hat{r}, \quad \forall t \in (0, \bar{t}]. \tag{4.28}$$

4. Let $\hat{r}$ be defined as above. If $d^* \ne 0$ and $x^{[k]}$ is feasible for problem (MONLP), then for all $\epsilon > 0$ there exists a $\bar{t} > 0$ such that

$$\bar{F}(x^{[k]} + td^*) - \epsilon\hat{r} \preceq_{\bar{F}} \bar{F}(x^{[k]}), \quad \forall t \in (0, \bar{t}]. \tag{4.29}$$

5. If $d^* = 0_{N_x}$, then $x^{[k]}$ is feasible for problem (MONLP) and there does not exist a $d \in C(x^{[k]})$ such that $\nabla_x F_l(x^{[k]})^\top d < 0$, $l = 1, \dots, N_F$, with the critical cone $C(x^{[k]})$ as in (2.12).

6. If $d^* \ne 0_{N_x}$, then there exists a $\bar{t} > 0$ such that for all $t \in (0, \bar{t}]$ and all $\sigma > 0$ sufficiently large, we have

$$\phi(x^{[k]} + td^*, \sigma) \le \phi(x^{[k]}, \sigma) + \mu t \nabla_x \phi(x^{[k]}, \sigma) d^*, \quad \forall \mu \in (0, 1). \tag{4.30}$$

Proof: See [46].

Note 4.7
Problem (4.25) can be seen as the simultaneous minimisation of the quadratic approximations of all objective functions, while feasibility is maintained.

Note 4.8
Problem (4.25) is the resulting QP from an intermediate iteration of an SQP method (see 2.1.2) for the problem

$$\begin{array}{lll} \min\limits_{x\in\mathbb{R}^{N_x}} & \sum_{l=1}^{N_F} F_l(x), & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, ..., N_g, \\ & h_j(x) = 0, & j = 1, ..., N_h. \end{array} \tag{4.31}$$

Performing SQP steps by solving problem (4.25) and taking points from a set $X \subset \mathbb{R}^{N_x}$ as initial guesses will result in a Pareto critical point. However, it may destroy the spread of any set X or will even result in a single point if problem (4.31) has a unique solution. The MOSQP algorithm [46] solves this problem by defining a reference point $\hat{x_k}$ for each point in X and replaces (4.25) with

$$\begin{array}{lll} \min\limits_{d\in\mathbb{R}^{N_x}} & \sum_{l=1}^{N_F} \nabla_x F_l(x^{[k]})^\top d + \frac{1}{2} d^\top H_l d, & \\ \text{subject to} & F_l(x^{[k]}) - F_l(\hat{x}^{[k]}) + \nabla_x F_l(x^{[k]})^\top d \leq 0, & l = 1, \ldots, N_F, \\ & g_i(x^{[k]}) + \nabla_x g_i(x^{[k]})^\top d \leq 0, & i = 1, ..., N_g, \\ & h_j(x^{[k]}) + \nabla_x h_j(x^{[k]})^\top d = 0, & j = 1, ..., N_h, \end{array} \tag{4.32}$$

and (4.31) respectively with

$$\begin{array}{lll} \min\limits_{x\in\mathbb{R}^{N_x}} & \sum_{l=1}^{N_F} F_l(x), & \\ \text{subject to} & F_l(x) \leq F_l(\hat{x_k}), & l = 1, \ldots, N_F, \\ & g_i(x) \leq 0, & i = 1, ..., N_g, \\ & h_j(x) = 0, & j = 1, ..., N_h. \end{array} \tag{4.33}$$

Fliege and Vaz also state that for any solution d^* to (4.32) the item 1 and items 3-6 of Lemma 4.6 hold [46].

Note 4.9
The additional constraints introduced in (4.33) can be seen as the worst-case acceptable outcomes of the optimisation problem considered. They are also quite similar to the additional constraints in the ε-constraints method (see Section 2.5.2) and introduce the risk of rendering the whole problem infeasible. In Figure 4.1 the situation is shown for constraints introduced with respect to infeasible reference points $\hat{x}_1^{[k]}$ and $\hat{x}_2^{[k]}$ which are both outside the feasible set displayed in blue. It can be seen that no point in the cone of $\hat{x}_1^{[k]}$ is feasible for problem (MONLP). However, although infeasible points $x^{[k]} \in X$ might lead to infeasible problems (4.33), it is not the case that this is true for all infeasible points as can be seen by the cone of $\hat{x}_2^{[k]}$.

As a solution to this problem, the idea is proposed here for points $x^{[k]}$ for which the solution of problem (4.33) is not successfully terminated. These points could be refined by finding a solution to the equation system defined by $g(x)$ and $h(x)$ without considering the objective functions. In this way, feasible points can be obtained, and the hope here is, since the solution to this equation system will typically not be unique, to find a new point close to the initial guess $x^{[k]}$ and thus retaining the spread of X.

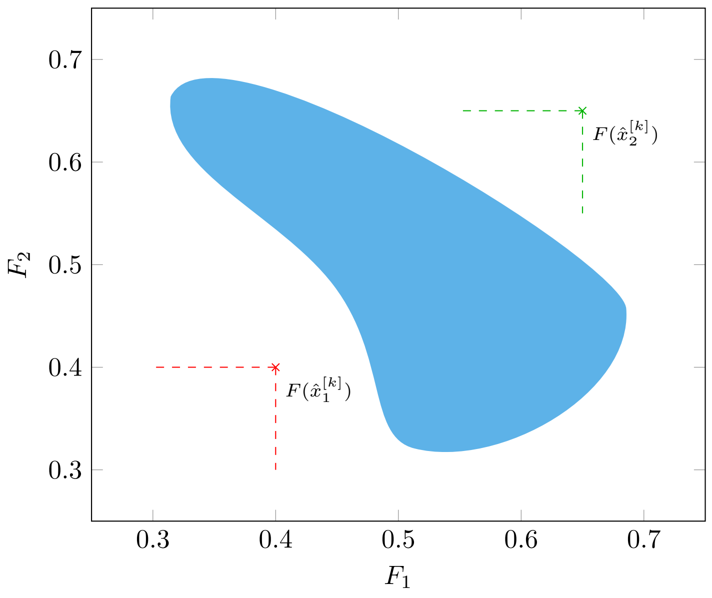


Figure 4.1: Pareto front with cone defined by different reference point $\hat{x}_k$. Dashed lines: Cones of infeasible point, Blue area: Feasible set.

This section demonstrates how search directions could be computed, which improve the spread of all points in a set X or drive those points toward local Pareto-optimality. Within the MOSQP algorithm, these two techniques are used by applying them in a parallel way to all points in a given set X and thus iterating the sets $X^{[k]}$. This iteration is done in two phases. In the first phase, the spread of the set is improved by computing SQP steps based on problem (4.14). In the second phase, the points are then refined based on problem (4.32). Following the idea of this thesis the points are, if necessary, subsequently processed for feasibility and again for local Pareto-optimality.

Before the final algorithm can be specified, there is one more problem to consider. In the spreading phase, the iterate sets $X^{[k]}$ grow exponentially since for every point in $X^{[k]}$ there are possibly N_F new points inserted into the set. A cleanup strategy was introduced in [46] to prevent the algorithm from using up all available memory in the computer. However, its original form has one major drawback, which is discussed in the next section.

4.2 Cleanup Strategies for Intermediate Approximations

To prevent the set of points from growing exponentially, a cleanup must be performed. This is done by selecting the most promising points after each iteration in the spread phase in the given implementation.

In general, it is important to evaluate how much a certain point improves the current approximation. The MOSQP algorithm [46] uses the crowding distance for this task. The crowding distance was introduced by Deb et al. [27] for the NSGA-II algorithm.
The idea is to evaluate the value of a solution based on the density of other solutions around it. To achieve this, the distance between the two neighbouring solutions is calculated for each objective function. In this way, it can be seen as the minimal cuboid in the neighbourhood of the point to be evaluated, which does not include another solution.

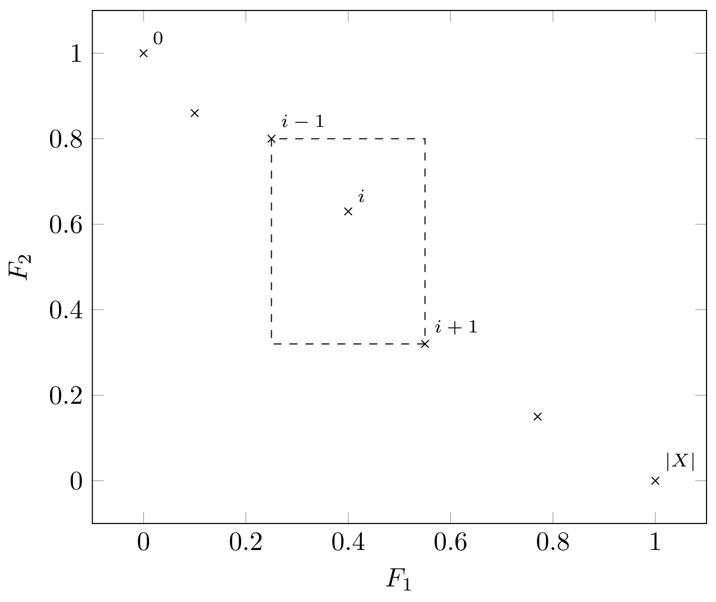


Figure 4.2: Illustration of the crowding distance [27].

The rough idea of the crowding distance is shown in Figure 4.2. The solution i is the point to be evaluated. The dashed line shows the minimal cuboid resulting from the distances of the two neighbouring solutions $i-1$ and $i+1$ with respect to f_1 and f_2. Looking at Figure 4.2, one interesting fact becomes clear, namely that the crowding distance does not depend on the exact objective values of the solution i but solely on the values of the neighbours. It must also be noted that the notion of neighbouring solutions is for dimensions $N_F > 2$ not as clear as in the picture. Therefore it is important to ensure that the neighbouring points are defined for only one objective at a time. This separation results in different neighbours for each objective. Deb et al. [27] state the following algorithm to assign a crowding distance to each point. Note that the indexing in the definition is a bit unusual, since it is more related to object-oriented programming where attributes like the distance d for the i-th point $X[i] \in X$ are assigned by $X[i]_{\text{distance}} = d$. However, it follows the original nomenclature in [27] and is also a very clear way of describing it.

Algorithm 4.10 (Crowding Distance)

1. Given set $X \subset \mathbb{R}^{N_x}$ as the current approximation.
2. Initialise $X[i]_{\text{distance}} = 0$ for all $i = 1, \dots, |X|$.
3. For each $l \in \{1, \dots, N_F\}$ do:

(i) Sort X regarding to objective l.

(ii) Set $X[1]_{\text{distance}} = X[|X|]_{\text{distance}} = \infty$.

(iii) For each $i \in \{2, \ldots, |X|-1\}$ do:

  (a) $X[i]_{\text{distance}} = X[i]_{\text{distance}} + (F_l(X[i+1]) - F_l(X[i-1]))$.

Here every point $X[i]$ is treated as a kind of container which includes not only the variable vector $x_i \in \mathbb{R}^{N_x}$ but also information about the solution like the crowding distance $X[i]_{\text{distance}}$

In the original algorithm in [46] the cleanup was performed in two steps.

Algorithm 4.11 (Pareto Based Cleanup)

1. Given set $X \subset \mathbb{R}^{N_x}$ and a number $0 < \bar{n} \in \mathbb{N}$.
2. Check for every point in X if it is dominated.
3. Select only the nondominated points in X and delete all others.
4. If $|X| > \bar{n}$ calculate the crowding distance (4.10) for all points in X and select the first $\bar{n}$ points with the highest crowding distance. Delete all other points.

Since the goal is to compute Pareto-optimal points, selecting only nondominated points for the next iteration is intuitive. However, there is one fundamental problem that leads to a worse approximation of the Pareto front.

The blue crosses in Figure 4.3 show a current approximation of an exemplary Pareto front within the algorithm. It can be seen that the outer two points dominate all remaining ones, even if they intuitively represent a good approximation.
The fact that this situation is not unlikely to occur when solving multi-objective optimisation problems can be seen in the figure. The blue line shows a highly concave Pareto front. The blue crosses are again the current approximation. Since feasible points are only located on the upper right side of the Pareto front, it is unlikely to find feasible points which are not dominated by the outermost ones in the early iterations of the algorithm. Furthermore, it is common to calculate the individual minima in the initialisation phase, which results in the two red points. In conclusion, in a situation as in Figure 4.3 it is very likely to find no nondominated points in early iterations. Applying a cleanup that only accepts nondominated would thus yield only two points after the cleanup.

This problem is not unknown in other algorithms, for example in heuristic approaches like the nondominated Sorting Genetic Algorithm (NSGA-II) [27]. This algorithm follows, on an abstract level, the same underlying idea as the MOSQP Algorithm, which can be described as two basic steps:

(I) Based on the current Approximation, generate new points.

(II) Filter the most promising points from the created points and delete the others.

The first step is very different in both approaches. This is where the strengths and weaknesses of the respective approaches lie. However, the idea of the second step is very similar. Thus it is possible to apply the selection method, the nondominated sorting, from the NSGA-II to the MOSQP Algorithm.

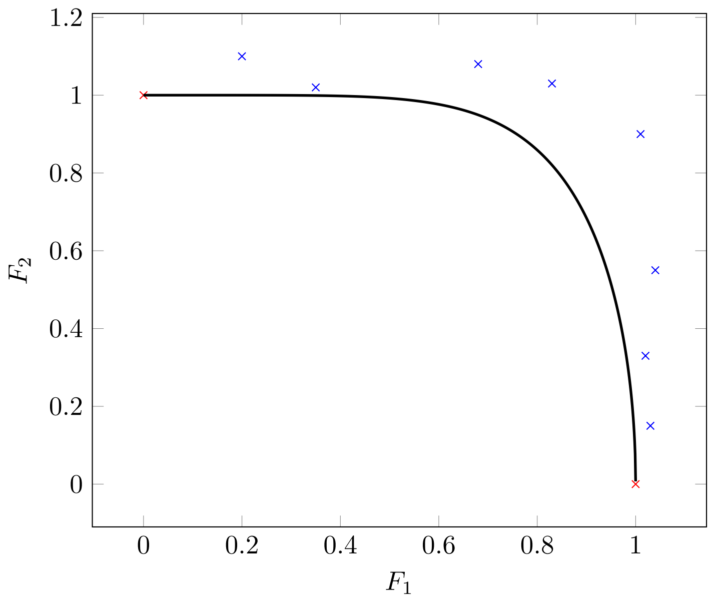


Figure 4.3: Pareto front with hypothetical dominated approximation. Red: Nondominated points, Blue: Dominated points, Black: Pareto front.

The basic idea of the selection method in the NSGA-II algorithm, called nondominated sorting, is that the points are not only divided into categories of dominated and nondominated but that different levels of "nondominatedness" are defined. This categorisation is done in two phases: first, order the points by their level of dominance and second, select based on this order:

Algorithm 4.12 (Nondominated Sorting)

(I) Given $X \subset \mathbb{R}^{N_x}$ and a number $0 < \bar{n} \in \mathbb{N}$.

(II) Assigning level of dominance:

- (i) Select the nondominated points in X and mark them as level 0.
- (ii) Set the current level $l = 1$.
- (iii) Select the nondominated points in X ignoring all points which already have a level assigned. Mark the selected points as level l.
- (iv) If a level has been assigned to all points: stop.
- (v) Set $l \leftarrow l + 1$ and continue with step (iii).

(III) Selection:

- (i) Set the current level $l = 0$, and $\emptyset = X_{\text{final}} \subset \mathbb{R}^{N_x}$.
- (ii) Set $X_l = \{x \in X \mid x \text{ has level } l\}$.
- (iii) If $|X_{\text{final}}| + |X_l| > \bar{n}$ got to step (vi).
- (iv) Set $X_{\text{final}} \leftarrow X_{\text{final}} \cup X_l$.
- (v) Set $l \leftarrow l + 1$ and go to step (ii).

(vi) Calculate the crowding distance (4.10) for all points in X_l and select the first $|X_{\text{final}}| - \bar{n}$ points with the highest crowding distance. Delete all other points in X_l and set $X_{\text{final}} \leftarrow X_{\text{final}} \cup X_l$.

The result of the nondominated sorting is shown in Figure 4.4. It can be seen that if the points in Level 1 are added to the nondominated ones, the approximation will likely yield much better results in the next iterations in the algorithm. However, Figure 4.4 also shows that even with nondominated sorting, the "better" points seem to be more concentrated near the individual minima. Thus it is not clear if the proposed cleanup strategy will result in a better Pareto front approximation. Nevertheless, one problem is solved, and that is the fact that a cleanup performed solely based on the division in dominated and nondominated points faces the risk that the number of points within the set will shrink over the iteration. In a setting like the one in Figure 4.3 it will even shrink to only two points within one iteration. If the two individual minima are computed within the initialisation phase, this will happen in the first iteration and thus make it very unlikely to compute a well-spread approximation. On the other hand, the nondominated sorting approach will guarantee that the current approximation will stay the same size over the iterations.

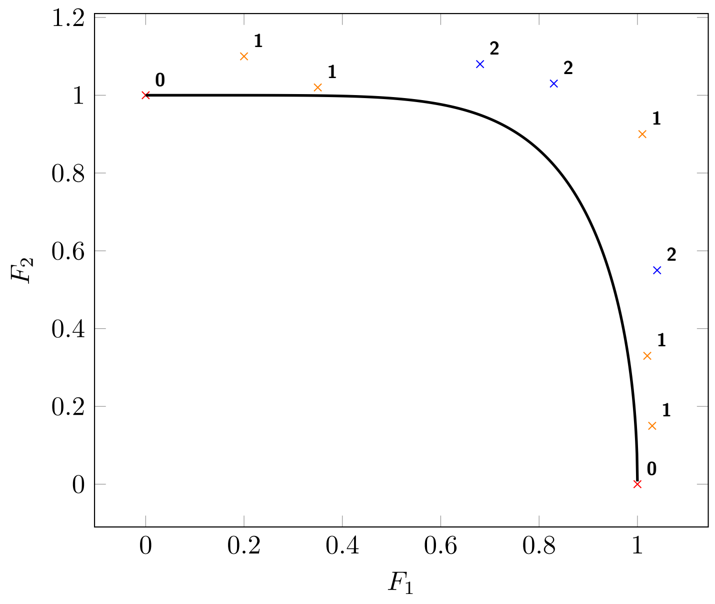


Figure 4.4: Pareto front with hypothetical approximation divided in levels.
Red: Level 0 (nondominated points), Orange: Level 1, Blue: Level 2,
Black: Pareto front.

4.3 Using WORHP in the MOSQP Algorithm

For the evaluation of the results of the original MOSQP algorithm, Fliege and Vaz [46] implement it using MATLAB routines for Quadratic Programming. This implementation a proper procedure, but it has the disadvantage of implementing the process of creating the QPs and the procedures for step size determination and regularisation of the Hesse matrix itself. This additional implementation effort wastes potential because the algorithm is based on SQP methods for which high-performance solvers have been available for some time.

This thesis focuses on implementing all necessary optimisation steps with WORHP (see Section 2.3 and [19]). Not only can the existing implementations for the computations mentioned above be used, but also many different methods like analytical calculations or finite differences to determine the derivative matrices can be chosen. Also, WORHP is a highly efficient solver which is designed to deal with large scale optimisation problems. This qualifies the interface for multi-objective optimisation developed in this thesis to handle e.g. discretised multi-objective optimal control problems. It also allows for the possibility of using parametric sensitivity analysis, a technique which is described in Section 2.2.1 and used, for example, in Chapter 3 in the context of Pareto front interpolation.

More details about WORHP can be found in Section 2.3 and in the respective literature [19]. Actual implementation details will be given in subsequent sections. At this point, it is important to know that the adapted MOSQP algorithm shown in the next section skips all steps concerning the step size or regularisation due to the usage of WORHP.

4.4 The Multi-Objective Sequential Quadratic Programming Algorithm

The complete MOSQP algorithm is described in 4.13 and the actual implementation is provided in [4]. Also Chapter 6 provides some details on the implementation.

Algorithm 4.13 (Multiobjective Sequential Quadratic programming (MOSQP))

```
(I) First Stage (Initialisation):
    Choose a nonempty, finite set of points $X^{[0]} \subset \mathbb{R}^{N_x}$. Let $\bar{n}$ be the number of points in $X^{[0]}$.
    Consider all points in $X^{[0]}$ as nonstopped. Chose a cleanup strategy $\mathcal{O}$. Set $k = 0$.

(II) Second stage (Spread):
    Perform a finite number of iterations on the set $X^{[k]}$ as follows.

    1. Set $X^{[k+1]} = \emptyset$.
       For each point in $x^{[k]} \in X^{[k]}$ not stopped do:
      1.(i) For each $l = 1, \ldots, N_F$ do:
        1.(i).a Compute the next point in direction of the $l$th objective $x^{[k+1,l]}$ that corresponds to the solution of one SQP step for objective function $l$ starting from $x^{[k]}$ by doing one iteration for the problem (4.17) with the SQP-Solver WORHP.
        1.(i).b If iteration was unsuccessful go to Step 1.(i).d.
        1.(i).c Set $X^{[k+1]} \leftarrow X^{[k+1]} \cup \{x^{[k+1,l]}\}$.
        1.(i).d Continue with the next $l$ .
      1.(ii) Set $x^{[k]}$ as a stopped point. Continue with the next point in $X^{[k]}$.
    2. Set all points in $X^{[k+1]}$ as nonstopped and perform the chosen cleanup strategy $\mathcal{O}$.
       Set $k \leftarrow k + 1$. If the pre specified finite number of iterations has not been reached and continue with step 1.
```

(III) Third stage (Optimality-refining stage):
For each point $x^{[k]} \in X^{[k]}$, set $\hat{x}^{[k]} := x^{[k]}$ as the reference point to be considered for problems (4.33).
Initialise the set $X^{[\text{final}]} = \emptyset$.

```
1. For all nonstopped points $x^{[k]} \in X^{[k]}$ do:
    1.(i)   Compute $x^{[k+1]}$ as the solution of problem (4.33) with a SQP Solver.
    1.(ii)  If the solver terminates successfully set $x^{[k+1]}$ as stopped, and set
            $X^{[\text{final}]} \leftarrow X^{[\text{final}]} \cup x^{[k+1]}$ else replace $x^{[k]}$ with $x^{[k+1]}$ in $X^{[k]}$.
    1.(iii) Continue with the next $x^{[k]}$.
2. If maximum number of iterations reached, or all points in $X^{[k]}$ are stopped, return
   $X^{[\text{final}]}$.
3. Set $X^{[k+1]} = \emptyset$.
   For each point in $x^{[k]} \in X^{[k]}$ not stopped do:
    3.(i)   Compute $x^{[k+1]}$ as a solution of the equation system formed by $g(x)$ and $h(x)$ by
            using the feasible only mode of WORHP.
    3.(ii)  $X^{[k+1]} \leftarrow X^{[k+1]} \cup \{x^{[k+1]}\}$.
4. Set $k \leftarrow k + 1$ and continue with Step 1.
```

4.5 Initialisation Strategies for General Multi-Objective Optimisation Problems

In the first phase, the MOSQP algorithm needs a non-empty finite set $X \subset \mathbb{R}^{N_x}$ as a seed for the spread phase. How this initial set is chosen has a significant impact on the performance of the solver. Thus, this section describes different approaches that will later be examined with regard to their respective advantages and disadvantages. On the one hand, the approaches investigated are effortless, while others are so advanced that they represent a kind of "pre-algorithm".
Some of these methods are taken from the original paper on the MOSQP algorithm. They are supplemented with strategies based on convex combinations between individual minima, which will later be examined with respect to their performance. However, some of the new methods are inspired by particular problems, and therefore their performance on general problems is relatively poor. Many of the presented methods also do not show any advantages in practice. However, since this knowledge is also useful, the methods are presented here. An evaluation of the practicability of the methods can be found in the Section 4.7.

Note 4.14
Some of the proposed initialisation strategies use the box bounds $x^{\text{lower}}, x^{\text{upper}} \in \mathbb{R}^{N_x}$ of a problem. These are not part of the general definition (MONLP) but are provided within the programming interface for the solver. A corresponding problem definition is given by (WORHP NLP).

Since the calculations are only well defined if all bounds are finite, Algorithm 4.21 only works for problems that are bounded in all variables. However, in praxis, WORHP replaces "unbounded" with a large number, usually chosen by hyperparameters. Thus they treat, for example, 10^{20} as unbounded, and all calculations become well defined again. This tweak is a bit unusual but provides a way to apply these strategies to a test set that contains unbounded problems.

Note 4.15
For some strategies the calculation of individual minima is necessary. However, the calculation of individual minima is a normal single-objective optimisation problem and thus has the same problems and preliminaries like the questions if the minimum exists, if it is unique and if it is a local or the global minimum. If no solution can be computed, all strategies based on these individual minima are usually not feasible. If a local minimum instead of the global one is found, the strategies can still be performed but will typically yield a worse spread.

4.5.1 User Provided Initialisation

The most basic method for initialisation is to delegate the responsibility to the user and consider a given set of points. This is not an initialisation strategy in the true sense of the word, but since the MOSQP algorithm offers this possibility [46], it should be mentioned here for the sake of completeness.

Within the unified WORHP interface described later in Chapter 6, the user can provide an initial guess for the optimisation variables x. These can also be used to get a point for the initial set X. However, this will only yield one single point and is a more special case of the User provided Initialisation. The procedure is mentioned because it works differently for the user as it uses an interface that is not MOSQP specific.

4.5.2 Random Gauss Initialisation

Around the user-provided initial guess, the set X can be enriched by inserting randomly generated points. In this case, the initial guess serves as the expected value of a normal distribution. To achieve a sufficiently wide distribution for all variables, the standard deviation for each variable is given as a percentage of the initial guess. Let $N(\mu, \sigma)$ denote the normal distribution with expected value μ and standard deviation σ where $r = N(\mu, \sigma)$ means r is one randomly computed number with distribution N.

Algorithm 4.16

```
1. Given the box bounds $x^{\text{lower}}, x^{\text{upper}} \in \mathbb{R}^{N_x}$ of a given problem, the corresponding initial
   guess $x^{\text{init}} \in \mathbb{R}^{N_x}$ feasible for the box bounds, a number $2 \leq \bar{n} \in \mathbb{N}$ of wanted points and
   the user defined value $p_x > 0$. Let $X^{[0]} \subset \mathbb{R}^{N_x}$ be the set of initialisation points initialised
   as $X^{[0]} = \emptyset$.
2. Compute for all $x_i$, $i = 1, \dots, N_x$ the wanted standard deviation $\sigma_i = x_i p_x$.
3. For $\ell = 0, \dots, \bar{n} - 1$ do:
     (i) For $i = 1, \dots, N_x$ do:
         (a) Compute $x_i^{[\ell]} = N(x_i^{\text{init}}, \sigma_i)$.
         (b) If $x_i^{[\ell]} < x_i^{\text{lower}}$ or $x_i^{[\ell]} > x_i^{\text{upper}}$ redo step (a).
         (c) Continue with next $i$.
    (ii) Set $X^{[\ell+1]} = X^{[\ell]} \cup \{x^{[\ell]}\}$.
   (iii) Continue with next $\ell$.
```

Note 4.17
The check in algorithm 4.16 step 3.(i).(b) should not lead to an infinite loop because even in the worst case $x^{\text{init}} = x^{\text{lower}}$ or $x^{\text{init}} = x^{\text{upper}}$ the expected value of recomputations is 2. For the implementation it is still a good idea to limit the number of iterations. If the maximum number of iterations is reached the value can be set to the bound value that is violated. This then leads to a slightly different distribution, but in practice this should not have much influence.
If better performance or a more precise distribution is needed it is also possible to use implementations of the truncated normal distribution [22].

4.5.3 Random Uniform Initialisation

The main goal of all initialisation methods is to find a set $X \subset \mathbb{R}^{N_x}$ of points that are well distributed within the objective space. In algorithm 4.16 the points are randomly created around a user-provided initial guess. This method has the disadvantage that the user also needs to provide the standard deviation σ for the distribution. A poorly chosen σ can result in an initial set X with too high a concentration by X. To circumvent this problem, the parameter σ could be set to a very large value. However, this would cause the expected value $\mu = x^{\text{init}}$ to lose its meaning. In such a situation, it would probably be more beneficial to switch from the Gauss distribution to a uniform distribution $U(l, u)$ which gives uniformly distributed points between $l \leq u \in \mathbb{R}$. As in the last section, box bounds of a form like (WORHP NLP) are used.

Algorithm 4.18

```
1. Given the box bounds $x^{\text{lower}}, x^{\text{upper}} \in \mathbb{R}^{N_x}$ of a given problem and a number $2 \leq \bar{n} \in \mathbb{N}$ of
   wanted points. Let $X^{[0]} \subset \mathbb{R}^{N_x}$ be the set of initialisation points initialised as $X^{[0]} = \emptyset$.
2. For $\ell = 0, \ldots, \bar{n} - 1$ do:
   (i) For $i = 1, \ldots, N_x$ do:
       (a) Compute $x_i^{[\ell]} = U(x_i^{\text{lower}}, x_i^{\text{upper}})$.
       (b) Continue with next $i$.
   (ii) Set $X^{[\ell+1]} = X^{[\ell]} \cup \{x^{[\ell]}\}$.
   (iii) Continue with next $\ell$.
```

4.5.4 Calculate Individual Minima

The method of calculating the individual minima before the actual algorithm often forms the basis for the further procedure in continuous nonlinear multi-objective optimisation. For example, the Normal Boundary Intersection method (NBI) depends on knowing the individual minima [26]. Since the individual minima are feasible and Pareto-optimal points, they often form a reasonable basis for generating further good points. However, the initial estimation containing nondominated points carries the risk that points obtained by other initialisation strategies will be dominated by them already in the first iteration. In the original MOSQP algorithm, this leads to the problem described in Section 4.2. The modification of the use of nondominated sorting proposed in this thesis solves this problem.

Algorithm 4.19

```
1. Given a Problem (MONLP). Let $X^{[0]} \subset \mathbb{R}^{N_x}$ be the set of initialisation points initialised
   as $X^{[0]} = \emptyset$.
   For $l = 1, \ldots, N_F$ do:
```

(i) Calculate the individual minima $x^{*[l]}$ of (MONLP) for the l-th objective function $F_l(x)$ as the solution of (WORHP NLP) with $f(x) \coloneqq F_l(x)$.

(ii) Set $X^{[l]} = X^{[l-1]} \cup \{x^{*[l]}\}$.

Note 4.20
In addition to the actual points added to the initial estimate by this method, the individual minima also form the basis for some of the more advanced initialisation strategies described below.

4.5.5 Line Initialisation between Bounds

One very simple idea of initialisation is to sample a line between the box bounds of the variables [46]. It can be seen as sampling the feasible set neglecting constraints other than the box bounds. Especially for problems that only have box constraints, this method can result in a very rich initialisation.

Algorithm 4.21

```
1. Given the box bounds $x^{\text{lower}}, x^{\text{upper}} \in \mathbb{R}^{N_x}$ for the variables $x \in \mathbb{R}^{N_x}$ of a given problem
   and a number $2 \leq \bar{n} \in \mathbb{N}$ of wanted points. Let $X^{[0]} \subset \mathbb{R}^{N_x}$ be the set of initialisation
   points initialised as $X^{[0]} = \emptyset$.
2. Compute the step size $s \in \mathbb{R}^{N_x}$ as $s = \frac{1}{\bar{n}-1}(x^{\text{upper}} - x^{\text{lower}})$.
3. For $\ell = 0, \ldots, \bar{n}-1$ do:
   (i) Compute $x^{[\ell]} = x^{\text{lower}} + \ell s$ and set $X^{[\ell+1]} = X^{[\ell]} \cup \{x^{[\ell]}\}$.
```

4.5.6 Line Initialisation between Individual Extrema for Bi-Objective Problems

Analogous to the idea of forming points on the line between the box constraints, such a line can also be formed between the individual minima of a bi-objective optimisation problem.

Algorithm 4.22

```
1. Given two individual minima $x^{*[l]}$ of problem (MONLP) with $N_F = 2$ obtained by algo-
   rithm 4.19 and a number $2 \leq \bar{n} \in \mathbb{N}$ of wanted points. Let $X^{[0]} \subset \mathbb{R}^{N_x}$ be the set of
   initialisation points initialised as $X^{[0]} = \emptyset$.
2. Compute the step size $s \in \mathbb{R}^{N_x}$ as $s = \frac{1}{\bar{n}-1}(x^{*[1]} - x^{*[2]})$.
3. For $\ell = 0, \ldots, \bar{n}-1$ do:
   (i) Compute $x^{[\ell]} = x^{\text{lower}} + \ell s$ and set $X^{[\ell+1]} = X^{[\ell]} \cup \{x^{[\ell]}\}$.
```

4.5.7 Generalised Initialisation between Individual Extrema

Algorithm 4.22 is only feasible for $N_F = 2$. However, the MOSQP algorithm itself is designed to work with problems where $N_F > 2$. Thus, it is important to have similar initialisation strategies for such problems.

The most natural generalisation of a line between two points to any number of points $x \in X \subset R^{N_x}$ is probably the general convex combination for weights $\omega \in \mathbb{R}^{|X|}$ with $\omega_l \geq 0$, for all $l = 1, \dots, |X|$:

$$x^c = \sum_{l=1}^{|X|} \omega_l x^{[l]}, \quad \text{where} \quad \sum_{l=1}^{|X|} \omega_l = 1, \quad x^{[l]} \in X. \tag{4.34}$$

In order to implement the initialisation with convex combinations, the question arises of how to determine the step size between adjacent weights ω such that the wanted number of points $2 \leq \bar{n} \in \mathbb{N}$ is matched. For a given step size s the number of weights can be calculated as

$$\frac{(m+p)!}{m!\, p!} = \binom{m+p}{p}, \tag{4.35}$$

where $n!$ represents the factorial of n, $p := |X| - 1$ and $m := \text{round}(\frac{1}{s})$. This equation can be concluded from the fact that the calculation of number of weights is equivalent to finding all possible combinations for p numbers which sum up to 1 which in turn is equivalent to finding all combinations to place the symbols "+" and "−" in a row where the number of "+" is equal to p and the number of "−" is equal to m. For example with $p = 5$ and $m = 6$,

$$+ - - + - + + - - + - \tag{4.36}$$

represents $\omega = (0, \frac{2}{6}, \frac{1}{6}, 0, \frac{2}{6}, \frac{1}{6})$. An explanation for this idea can also be found at [121]. Since m is defined by the rounded reciprocal of s, it becomes obvious that not all stepsizes are possible. In addition, by varying m and p it is only possible to compute the subset of $\mathbb{N}$ which is given by all possible binomial coefficients. Thus, it is not possible to find a specific stepsize s for all given $\bar{n}$. Two possible substitutes can be considered: First the biggest number $m^{[b]} \in \mathbb{N}$ for which

$$\binom{m^{[b]}+p}{p} \leq \bar{n}, \tag{4.37}$$

and second the smallest number $m^{[s]} \in \mathbb{N}$ for which

$$\binom{m^{[s]}+p}{p} \geq \bar{n}. \tag{4.38}$$

Since m and p are still relatively small in the context of multi-objective optimisation, thus resulting in low computational effort for computing all weights, and in addition, the MOSQP Algorithm strongly benefits from a rich initial guess for X, this thesis proposes an implementation which chooses $m^{[s]}$ for the calculations within the initialisation phase, resulting in the following algorithm

Algorithm 4.23

1. Given the individual minima $x^{*[l]}$, $l = 1, \dots, N_F$ of problem (MONLP) with $N_F \geq 2$ obtained by Algorithm 4.19 and a number $2 \leq \bar{n} \in \mathbb{N}$ of wanted points. Let $X^{[0]} \subset \mathbb{R}^{N_x}$ be the set of initialisation points initialised as $X^{[0]} = \emptyset$.

2. Compute the step size $s \in \mathbb{R}^{N_x}$ as $s = \frac{1}{m^{[s]}}$.

3. Compute set of all weights

$$\Omega = \left\{ \omega \in \mathbb{R}^{N_F} \mid \sum_{l=1}^{N_F} \omega_l = 1, \text{ and } \omega_l = zl \text{ for } z \in \mathbb{N}_0 \right\}.$$

4. For $\ell = 1, \ldots, |\Omega|$ do:
   (i) Compute $x^{[\ell]} = \sum\limits_{l=1}^{N_F} \omega_l x^{*[l]}$ and set $X^{[0]} \leftarrow X^{[0]} \cup \{x^{[\ell]}\}$.

The minimum which is defined in step 3 in essence describes the stepsize between two weight vectors.

4.5.8 Convex Combination Initialisation between Individual Extrema with Feasibility Refinement

For problems with a high number of nonlinear constraints, such as discretised optimal control problems, it is sometimes beneficial to start with a feasible initial guess. In order to provide such an initial guess $X^{[0]}$ which is also well spread in the objective space, the idea is to refine all points $X^{[0]}$ obtained by algorithm 4.23 for feasibility. For this purpose some steps of the refinement phase in the MOSQP Algorithm 4.13.(III) are applied to the points in $X^{[0]}$.

Algorithm 4.24

1. Given points $X^{[0]} \subset \mathbb{R}^{N_x}$.
2. Do step (III).3. of the MOSQP Algorithm 4.13 with $k = 0$.

4.5.9 Convex Combination Initialisation between Individual Extrema with Reference Point Refinement

Algorithm 4.24 provides feasible solutions. However, these refinements can destroy the spread within the objective space since they do not consider the objective values when applied to points in $X^{[0]}$ that are obtained by Algorithm 4.23. Thus, the goal could be reformulated as not only finding feasible points, but to find feasible points that are as close to the points in $Y^{[0]} = \{y \mid y = F(x) \text{ with } x \in X^{[0]}\}$ as possible, which can be described as

$$\begin{array}{lll} \min\limits_{x} & \frac{1}{N_F} \sum\limits_{l=1}^{N_F} ||F_l(x) - y^{\text{ref}}||^2, & \\ \text{subject to} & g_i(x) \leq 0, & i = 1, \ldots, N_g, \\ & h_j(x) = 0, & j = 1, \ldots, N_h, \end{array} \tag{4.39}$$

with $y^{[\text{ref}]} \in Y^{[0]}$.

Algorithm 4.25

1. Given points $X^{[0]} \subset \mathbb{R}^{N_x}$.
2. For all points $x^{[\ell]} \in X^{[0]}$ do:
   (i) Compute $x^{*[\ell]}$ as the solution of (4.39) with $y^{\text{ref}} := F(x^{[\ell]})$.

(ii) Replace $x^{[\ell]}$ with $x^{*[\ell]}$ in $X^{[0]}$.

Note 4.26
Algorithms 4.24 and 4.25 contain quite expensive computations, which makes them very heavy-weight compared to pure initialisation algorithms, especially since the MOSQP algorithm has significantly fewer problems in the treatment of constraints than heuristic approaches, due to its use of the SQP technique based on the Lagrange function.
However, since there is no need for high-quality solutions, it is also possible to lower tolerances and the maximal number of iterations to reduce computational effort.

Note 4.27
The strategies shown can also be applied solely to the control variables when dealing with discretised optimal control problems. If all state values are set for the first discrete point, the initial guess for discretised states can be computed by integrating the ode system based on these controls. In this way, the a-priori known connection between discrete points is used while still getting some spread within the initial guess.

4.6 Basic Terms for the Numerical Evaluation of the Algorithms

4.6.1 Test Problems

The test problems for the evaluation of different solvers are mainly taken from the problem compilation in Thomann [114]. Those problems represent a large variety of different shapes of the Pareto front. It represents convex and non convex shapes, as well as connected and unconnected ones. However, it consists solely of problems with only box constraints or even unbounded problems. No general constraints are represented. In general, this is not a big problem since the evaluation of solvers is mainly about their coverage of the Pareto front. Still, to get a more accurate evaluation, the set of test problems is enriched with problems from the compilation in Fliege and Vaz [46]. Table 4.1 provides an overview of the problems with their respective dimensions for the number of objectives N_F, variables N_x and general constraints N_g. The implementation of the test set can be found in [4].

4.6.2 Performance Profiles

To compare two or more solvers, it is essential to define what exactly is "good" or "better". In the field of numerical optimisation, the so-called Performance Profiles are a standard tool for this purpose. These have the advantage of providing a measure of how good a solver is concerning a specific criterion and they represent a kind of robustness. For this, they are defined by a cumulative function $\rho(\tau)$ which represents a performance ratio concerning a given metric for a given set of solvers [46].

Let the set $\mathcal{SO}$ of all solvers be given, $\mathbf{P}$ the set of all problems under consideration. Furthermore let $t^{[p,s]}$ be the performance of solver s on solving problem p. Define the performance ratio as

$$r^{[p,s]} := \frac{t^{[p,s]}}{\min_{\bar{s}\in\mathcal{SO}} t^{[p,\bar{s}]}}. \tag{4.40}$$

Problem	N_F	N_x	N_g	Problem	N_F	N_x	N_g	Problem	N_F	N_x	N_g
BK1	2	2	0	Lis	2	2	0	MOQP_0001	3	20	20
CL1	2	4	0	lovison1	2	2	0	MOQP_0002	3	20	20
Deb41	2	2	0	lovison2	2	2	0	MOQP_0003	3	20	20
Deb53	2	2	0	lovison3	2	2	0	OF1	2	2	1
Deb513	2	2	0	lovison4	2	2	0	TR1	2	2	2
Deb521b	2	2	0	MOP1	2	1	0	WR1	2	5	2
DG01	2	1	0	Schaffer2	2	1	0	ABCcomp	2	3	2
DLTZ1	2	2	0	T1	2	2	0	DD1	2	5	3
ex005	2	2	0	T2	2	2	0	FDS	3	100	0
Far1	2	2	0	T3	2	2	0	GE1	2	2	1
FES2	3	10	0	T4	2	50	0	GE2	2	40	0
FF	2	2	0	T5	2	2	0	GE3	2	2	2
Fonseca	2	2	0	T6	2	2	0	GE4	3	3	1
IKK1	3	2	0	T7	2	3	0	GE5	3	3	0
IM1	2	2	0	T8	3	3	0	HM1	2	2	0
Jin1	2	50	0	VU1	2	2	0	HS5	3	5	3
Jin2	2	5	0	VU2	2	2	0	Jo1	2	5	3
Jin3	2	5	0	ZDT1	2	4	0	liswetm	2	7	5
Jin4	2	5	0	ZDT2	2	4	0	MOLPg_001	3	8	8
JOS3	2	3	0	ZDT3	2	4	0	MOLPg_002	3	12	12
Kursawe	2	3	0	ZDT4	2	4	0	MOLPg_003	3	10	10
Laumanns	2	2	0	ZDT6	2	4	0				
LE1	2	2	0	ZLT1	3	4	0				

Table 4.1: Overview of test problems.

For solver $s \in \mathcal{SO}$ define the function

$$\rho^{[s]}(\tau) := \frac{\left|\left\{p \in \mathbf{P} : r^{[p,s]} \leq \tau\right\}\right|}{|\mathbf{P}|}, \tag{4.41}$$

as the percentage of problems with a performance below or equal to τ. The value of $\rho^{[s]}(\tau)$ can be interpreted as the probability of the performance of s on an arbitrary problem to be within a factor of τ of the best performance. By comparing the values of $\rho^{[s]}(1)$ for each solver, the most efficient solver can be determined, whereas the graph of $\rho^{[s]}(\tau)$ for $\tau \to \infty$ gives information about the robustness of solver s, since larger $\rho^{[s]}(\tau)$ for large τ means solver s is able to solve a larger number of problems from $\mathbf{P}$.

In order to calculate performance profiles, the underlying metrics for $t^{[p,s]}$ have to be defined. This is usually an algorithmic performance metric like the number of iterations a solver needs to solve a problem. In multi-objective optimisation, these metrics pose a particular challenge since the output of a solver is usually a set of points, which then has to be compared to the sets coming from other solvers. How to measure the quality of those sets is widely discussed in the literature. For all comparisons in this thesis the purity metric [24], two spread metrics [24] and the hypervolume metric [127] are used.

The Purity Metric

The purity gives a measure of how good a solver is at finding nondominated points in it. Ideally, the points found are compared with the true Pareto front. If they lie on this, they are not dominated points. However, since the true Pareto front is not known, the set of all solutions found by all solvers is compared instead. For this, let $\mathcal{P}^{[p,s]}$ be the approximation to the Pareto front computed by solver s. The substitution for the true Pareto front is defined by $\mathcal{P}^{[p]} := \bigcup_{s \in \mathcal{SO}} \mathcal{P}^{[p,s]}$. All dominated points are removed from $\mathcal{P}^{[p]}$. The purity metric is then defined by the points in $\mathcal{P}^{[p]}$ divided by the number of nondominated points a certain solver $s$can compute, resulting in

$$t^{[p,s]} := \begin{cases} \frac{\left|\mathcal{P}^{[p]}\right|}{\left|\mathcal{P}^{[p,s]} \cap \mathcal{P}^{[p]}\right|}, & \left|\mathcal{P}^{[p,s]} \cap \mathcal{P}^{[p]}\right| \neq 0, \\ \infty, & \left|\mathcal{P}^{[p,s]} \cap \mathcal{P}^{[p]}\right| = 0. \end{cases} \tag{4.42}$$

As required for performance profiles, small values of (4.42) are better than high values, and a value $t^{[p,s]} = \infty$ means that the solver was not able to find one single non dominated point. More details can be found in [24].

The Spread Metrics

Besides the information about how many nondominated points a solver can find, it is also essential to know how well they are distributed. The central idea here is that the computed points each represent as much information as possible about the Pareto front. As an example, this means that if many points are found in a very dense space, they contain little information about the Pareto front, even if they are not dominated. In contrast, even a few points that are "well distributed" can tell a lot about the general shape. For example, in the case of simple fronts, three well-distributed points are enough to tell whether the Pareto front is convex or not. Two different spread metrics are given in the literature to measure how "well distributed" a set of points is.

In order to define the spread metrics it is assumed that the approximation found by solver s for problem p consists of $\bar{n}$ points $x_1, \dots, x_{\bar{n}}$. In the following definitions it is also assumed that, when considering the objective function l, the points are ordered with respect to this objective function l, i.e. $F_l(x_\ell) \leq F_l(x_{\ell+1}) (\ell = 1, \dots, \bar{n} - 1)$. Additionally, for the following considerations the extreme points for objective function l are denoted by x_0 and $x_{\bar{n}+1}$ in the respective definitions; i.e., x_0 represents the global minimum of F_l and $x_{\bar{n}+1}$ the global maximum of F_l. In practical evaluations, these values are of course not known with complete confidence. Therefore, the best known points for x_0 and $x_{\bar{n}+1}$ from the approximations of all solvers used are used. Now, define

$$\delta_{l,\ell} := |F_l(x_{\ell+1} - F_l(x_\ell))| \quad \text{and} \tag{4.43}$$

$$\bar{\delta}_l := \frac{1}{\bar{n}+1} \sum_{\ell=0}^{\bar{n}} \delta_{l,\ell}, \quad l = 1, \dots, N_F. \tag{4.44}$$

The two spread metrics $\Gamma > 0$ and $\Delta > 0$ are then defined as

$$\Gamma^{[p,s]} := \max_{l \in \{1,\dots,N_F\}} \max_{\ell \in \{0,\dots,\bar{n}\}} \delta_{l,\ell}, \tag{4.45}$$

and

$$\Delta^{[p,s]} := \max_{l \in \{1,\dots,N_F\}} \left(\frac{\delta_{l,0} + \delta_{l,\bar{n}} + \sum_{\ell=1}^{\bar{n}} \left|\delta_{l,\ell} - \bar{\delta}_l\right|}{\delta_{l,0} + \delta_{l,\bar{n}} + (\bar{n}-1)\bar{\delta}_l} \right). \tag{4.46}$$

Including x_0 and $x_{\bar{n}+1}$ in the definition leads to the fact that not only the distribution within the respective approximation is considered, but the distribution over the entire Pareto front. This information is important because a solver may find points with good distribution in themselves but a large distance to the extreme points. The Γ metric measures the largest gap in the Pareto front, the Δ metric measures the scaled deviation from the average gap in the Pareto front. For more details see [24].

The Hypervolume Metric (S-Metric)

The hypervolume metric is, roughly speaking, an indicator of how much space a given approximation dominates. It represents in one single value information how well spread an approximation is and additionally how far away it is from the actual Pareto front. It also has the property that the maximal possible value can only be achieved by finding an approximation that covers the whole Pareto front.
Despite this simple description of the properties, calculating the hypervolume of a given approximation is not trivial, and many algorithms exist for this. This thesis uses the algorithm which can be found under the web address given in [123]. An interface was developed for this purpose directly from the main program. For comprehensive information see [13].
The hypervolume is for an approximation consisting of only one point 0. To avoid a division by 0 in the context of the performance profile, this work uses the modification proposed in [118] in which all values are brought into the positive range if necessary:

$$t^{[p,s]} := \begin{cases} \tilde{t}^{[p,s]} + 1 - \min_{\bar{s}\in\mathcal{SO}} \tilde{t}^{[p,\bar{s}]}, & \min_{\bar{s}\in\mathcal{SO}} \tilde{t}^{[p,\bar{s}]} < \varepsilon, \\ \tilde{t}^{[p,s]}, & \text{otherwise,} \end{cases} \tag{4.47}$$

where $\tilde{t}^{[p,s]}$ represents the hypervolume for the approximation of solver s for problem p. In this thesis $\varepsilon = 0.001$ is used.

4.6.3 Data Profiles

In addition to the performance profiles, there also exist data profiles. Where performance profiles evaluate the quality of an approximation, data profiles measure how much resources, i.e. function evaluations, an algorithm needs to solve a problem. They were introduced by [93] in order to compare derivative-free single-objective optimisation algorithms. In [24] they were adapted for multi-objective optimisation. For any value $0 < \sigma \in \mathbb{R}$ they are defined as

$$d^{[s]}(\sigma) := \frac{\left|\left\{p \in \mathcal{P} : h^{[p,s]} \leq \sigma\right\}\right|}{|\mathcal{P}|}, \tag{4.48}$$

where $h^{[p,s]}$ is the number of function and derivative evaluations solver s needs to solve problem p.
One problem with this definition in the context of multi-objective optimisation is the question of what exactly "solved" means. In order to answer this question, it must first be clarified in more detail what the data profiles are to evaluate. That is: How much resources does an algorithm need in order to converge. The original version in [93] uses the best known solution of the problem for this purpose. In [24] that formulation is adapted and the notion of "solved" is defined as follows.
Let $\mathcal{P}^{[p]}$ be a set of points that we consider a high-quality approximation of the Pareto front. Then, a solver s is said to solve problem p with accuracy ε, if

$$\frac{\left|\mathcal{P}^{[p,s]} \cap \mathcal{P}^{[p]}\right|}{\left|\mathcal{P}^{[p]}\right| / |\mathcal{SO}|} \geq 1 - \varepsilon, \tag{4.49}$$

so that the relative proportion of points obtained in the reference Pareto front $\mathcal{P}^{[p]}$ is equal to or greater $1 - \varepsilon$. Note that in (4.49) the number of points in $\mathcal{P}^{[p]}$ is divided by the number of solvers in $\mathcal{SO}$. The reason behind this is that it is assumed that all solvers contribute equally to

the reference Pareto front.
In this scenario, $\mathcal{P}^{[p]}$ is computed by initially letting all solvers in $\mathcal{SO}$ run for 5000 function evaluations and then merging the obtained approximations and filtering all dominated points out of the resulting set.
This formulation however has some drawbacks. First of all the division by $\left|\mathcal{P}^{[p]}\right| / |\mathcal{SO}|$ is not fair if not all solvers provide the same number of points to the set $\mathcal{P}^{[p]}$. This can be the case if one solver computes dominated points. It can be argued that a solver is not really good if it is not able to compute nondominated points with such a high number of iterations. However, this ability is already judged by the purity metric (4.42) and should therefore not be additionally rated here. For these reasons it is proposed here to replace $\left|\mathcal{P}^{[p]}\right| / |\mathcal{SO}|$ by $|\mathcal{P}^{[p,s]}|$ and thus to measure the percentage of a solution $|\mathcal{P}^{[p,s]}|$ which is on the Pareto front $\mathcal{P}^{[p]}$. In this way (4.49) becomes

$$\frac{\left|\mathcal{P}^{[p,s]} \cap \mathcal{P}^{[p]}\right|}{\left|\mathcal{P}^{[p,s]}\right|} \geq 1 - \varepsilon. \tag{4.50}$$

The second thing to notice about (4.49) is that the fact if a point $x \in \mathcal{P}^{[p,s]}$ is on the Pareto front is determined by whether the point is in the intersection with $\mathcal{P}^{[p]}$. If now a heuristic like the NSGA-II is rated, which is inherently based on random processes and thus might find a point which is nondominated but not the same as was found before, the point is discarded at not solving the problem. In order to overcome this problem with the randomness of the NSGA-II this thesis proposes two different approaches. The first is to consider a point $x \in \mathcal{P}^{[p,s]}$ to be "on" the Pareto front if

$$\mathcal{P}^{[p,s]} \cap_d \mathcal{P}^{[p]} := \left\{ x \in \mathcal{P}^{[p,s]} | \exists \tilde{x} \in \mathcal{P}^{[p]} : |F_l(x) - F_l(\tilde{x})| < 0.5\bar{\delta}_l \ \forall \ l = 1, \ldots, N_F, \right\}, \tag{4.51}$$

with $\bar{\delta}_l$ as in (4.44) computed for $\mathcal{P}^{[p]}$. The idea is now to replace the intersection by all points in $x \in \mathcal{P}^{[p,s]}$ which satisfy (4.51), which is then denoted by $\mathcal{P}^{[p,s]} \cap_{\bar{\delta}} \mathcal{P}^{[p]}$. Figure 4.5 shows the idea. The area spanned by $\bar{\delta}$ is displayed as the dashed line. It can be seen, that the point x_1 in blue is here considered to be within the intersection $\mathcal{P}^{[p,s]} \cap_{\bar{\delta}} \mathcal{P}^{[p]}$. However, the obvious drawback of this method is that also the point x_2 is now considered to be part of the solution, even though it is dominated by $\tilde{x}$. However, in the original definition of data profiles in [93] it was also the idea to not require the solution under examination to be equal to the best known solution. Thus the defintion (4.51) could be considered an adaption of this idea to multi-objective optimisation.

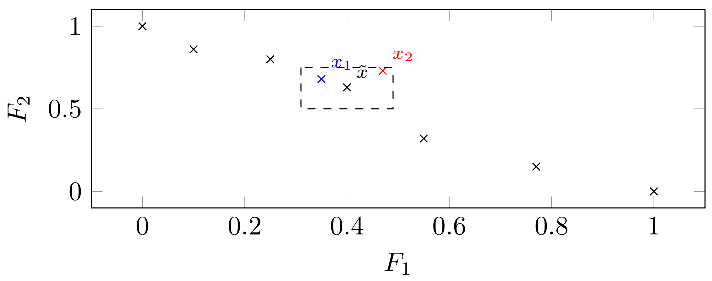


Figure 4.5: Illustration of intersection based on gap size (4.51).

However, one other idea could also be seen as a natural adaption to multi-objective optimisation, namely to simply consider all points $x \in \mathcal{P}^{[p,s]}$ which are not dominated by any point

$\tilde{x} \in \mathcal{P}^{[p]}$ to be part of the solution, which is here defined as

$$\mathcal{P}^{[p,s]} \cap_{\leq} \mathcal{P}^{[p]} := \left\{ x \in \mathcal{P}^{[p,s]} \mid \nexists \tilde{x} \in \mathcal{P}^{[p]} \text{ with } \tilde{x} < x \right\}. \tag{4.52}$$

Since all three “intersections” ($\cap, \cap_{\bar{\delta}}, \cap_{\leq}$) have their respective advantages and disadvantages they are all considered within this thesis and the respective data profiles are denoted with $\tilde{d}, d^s_{\cap}, d^s_{\cap_{\bar{\delta}}}, d^s_{\cap_{\leq}}$ where $\tilde{d}$ is the original data profile and d is the one where only the points of each solver itself are considered.

4.7 Comparison of the NsSQP and MOSQP Algorithm

In this section, a numerical comparison of the effects of the modification of the MOSQP algorithm is presented. In the following, a distinction is made between the Multiobjective SQP (MOSQP) algorithm and the Nondominated Sorting SQP (NsSQP) algorithm, whereby the former represents the original variant with pure dominance as a selection criterion (see Algorithm 4.11) and the latter the modified variant in which the nondominated sorting of the NSGA-II is used (see Algorithm 4.12).

For the purpose of comparison, both variants are first compared with different initial-guess methods using the performance (see Section 4.6.2) and data metrics (see Section 4.6.3). The implementation is provided in [4].

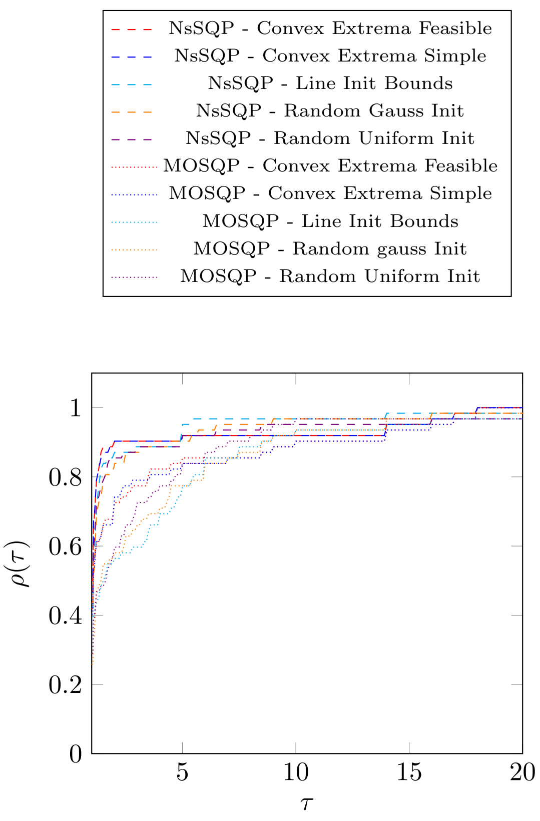
4.7.1 Comparison of Performance Profiles for the MOSQP and NsSQP Algorithm

For the comparison of the MOSQP and the NsSQP algorithm the whole test set is considered in this section. Both algorithms are run with the initialisation methods “Random Gauss Init” (Alg. 4.16), “Random Uniform Init” (Algorithm 4.18), “Line Init Bounds” (Algorithm 4.21), “Convex Init Extrema Feasible” (Algorithm 4.24) and “Convex Init Extrema Simple” (Algorithm 4.23). Also the following parameters were used:

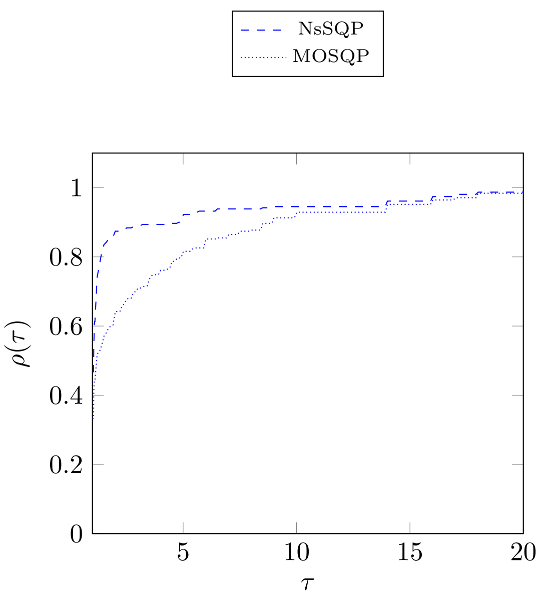
Maximal Points in Set	20
Maximal iterations in Spread Phase	20
Maximal iterations in Refine Phase	20

To show the difference between the two algorithms, two plots for each performance metric are shown. The first displays the individual performance metric plots for both algorithms and each initialisation strategy. The second displays the average of all initialisation strategies for both algorithms. The average is computed for each value of τ for the values of $\rho^{[s]}(\tau)$. This way, it becomes clear if one of the versions performs better on average for different initialisation sets. At the same time, it becomes clear how high the variance within the performance metrics can be if the initialisation is varied.

First, the algorithms are examined using the purity metric (4.42). Figure 4.6a shows the performance graph for each variant of the MOSQP algorithm as dotted lines and the NsSQP as dashed lines. For each initialisation strategy, the same colour is used for both algorithms. Thus, the first thing to notice is that all graphs belonging to the NsSQP algorithm are almost completely above all graphs of the MOSQP algorithm. Supplementing this observation are the graphs for the average values in Figure 4.6b. This would support the claim that the usage of



(a) Individual solver performances

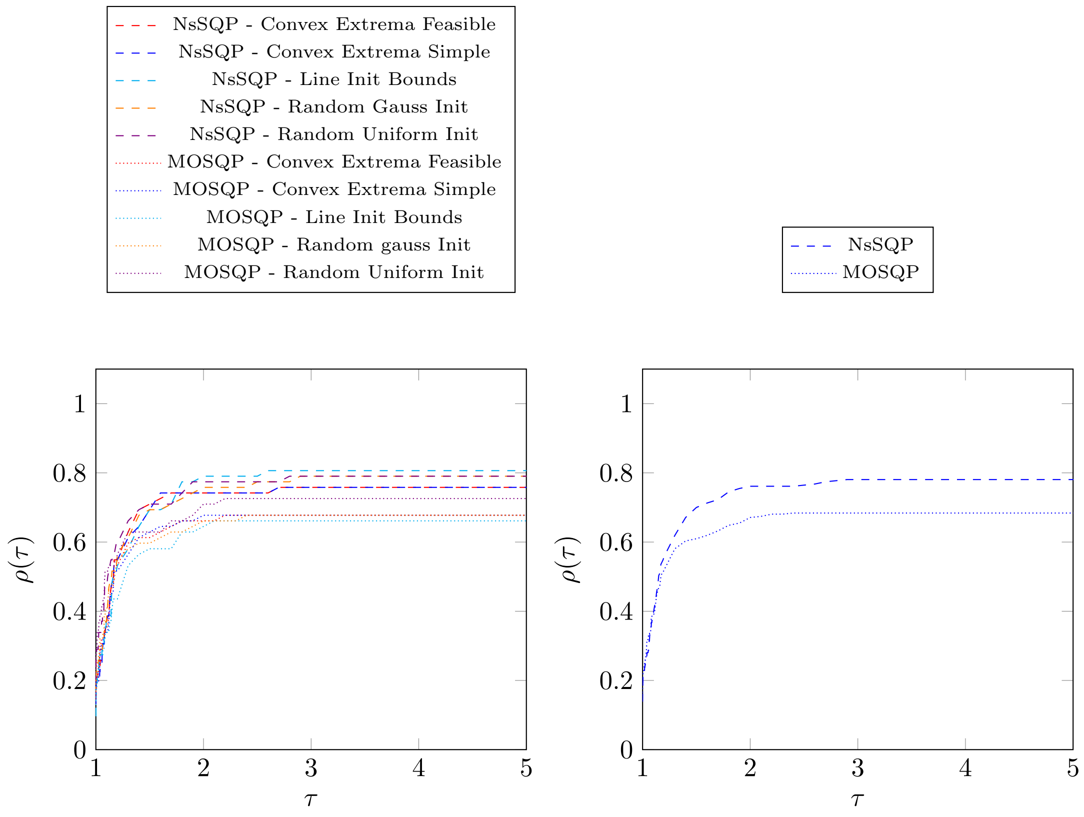


(b) Averages of the solver

Figure 4.6: Performance profile for MOSQP and NsSQP for the purity metric.

nondominated sorting enables the algorithm to compute more nondominated points. Especially the fact that the average graph of the NsSQP algorithm rises much faster at the beginning shows that the NsSQP performs better in terms of the purity metric.
Another interesting observation is that the relative order of the initialisation strategies is roughly the same for the MOSQP and NsSQP algorithms. Both strategies based on convex combinations of the extreme points perform the best in both algorithms. In addition, these are also the two versions where the difference between both algorithms is the highest. One reason for this is probably that the single-objective extrema included in the initialisation strategies “Convex Extrema Feasible” and “Convex Extrema Simple” are already nondominated points. Including these points makes it harder to find other points, which are not dominated by one of the extrema within the initialisation, thus leading to precisely the effect described in the motivation to use nondominated sorting in Section 4.2.
Overall, the influence of the chosen initialisation strategy is in the same magnitude as the influence of the chosen algorithm version. However, regarding the purity metric, the NsSQP algorithm is a clear improvement compared to the MOSQP in all versions.

At first glance, the graphs regarding the spread metric “Deviation” in Figures 4.7a and 4.7b seem very similar to those of the purity metric. Again, the average graph in Figure 4.7b of the

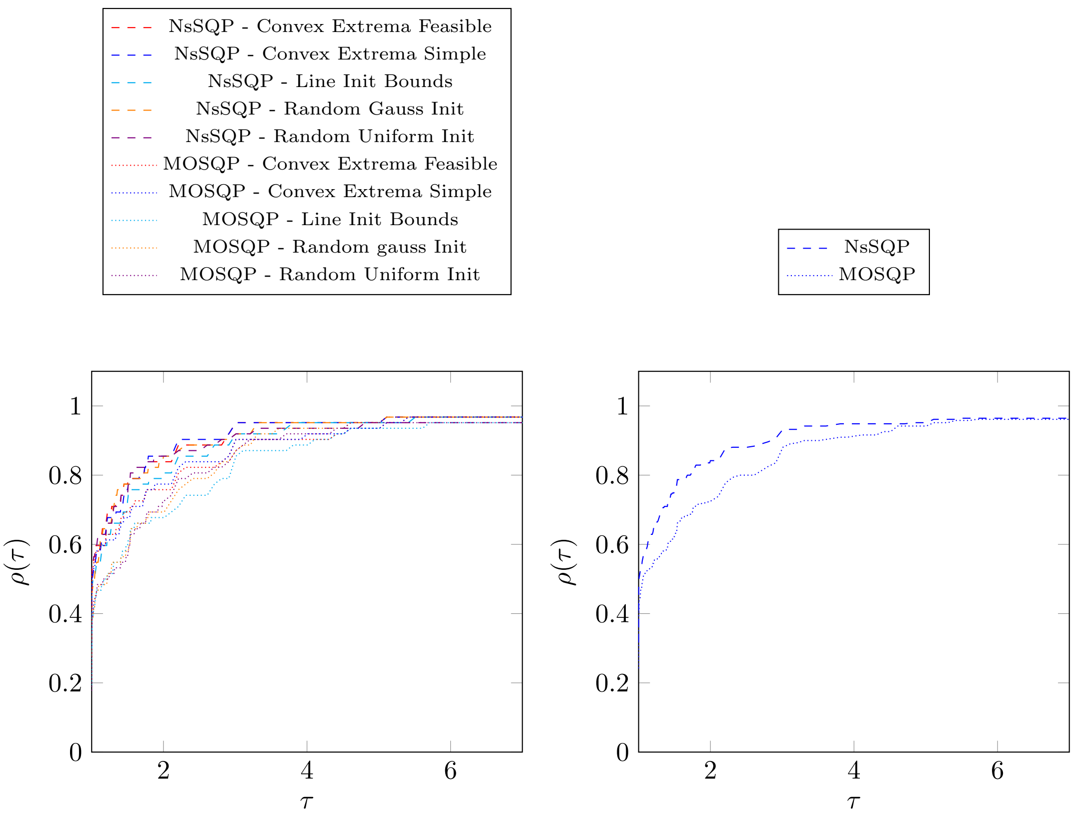


(a) Individual solver performances

(b) Averages of the solver

Figure 4.7: Performance profile for MOSQP and NsSQP for the Δ deviation - spread metric.

NsSQP algorithm runs above the one of the MOSQP algorithm just like the graphs in Figure 4.7a. However, a closer look reveals some differences in the details. The first difference is the significantly lower variance in the curves, both between the algorithms and between the individual initialisation strategies. Especially the initialisation strategies seem to have no notable influence on the Δ metric.
Also, the graphs of the NsSQP algorithm in Figure 4.7a are not as strictly above those of the MOSQP algorithm as in the purity metric. For values $\tau < 1.5$, the performance of the MOSQP algorithm with the strategy "Random Uniform Init" is the highest, even if only marginal. There are hardly any differences between the various methods in this area. The NsSQP algorithm shows higher values only in the range from $\tau > 1.5$, which hints at a better robustness of the NsSQP algorithm regarding the spread metric Δ.
Finally, it should be noted that the relative order of the individual initialisation strategies in the delta metric is not the same for NsSQP and MOSQP compared to the purity metric. Instead, the worst MOSQP strategy, "Line Init Bounds" (shown in cyan), is the best for the NsSQP algorithm. However, it should be noted that since the variance of the methods within the same algorithm is small, this fact is not of high relevance. Overall it can again be observed that the NsSQP approach improves the MOSQP approach for each initialisation method.



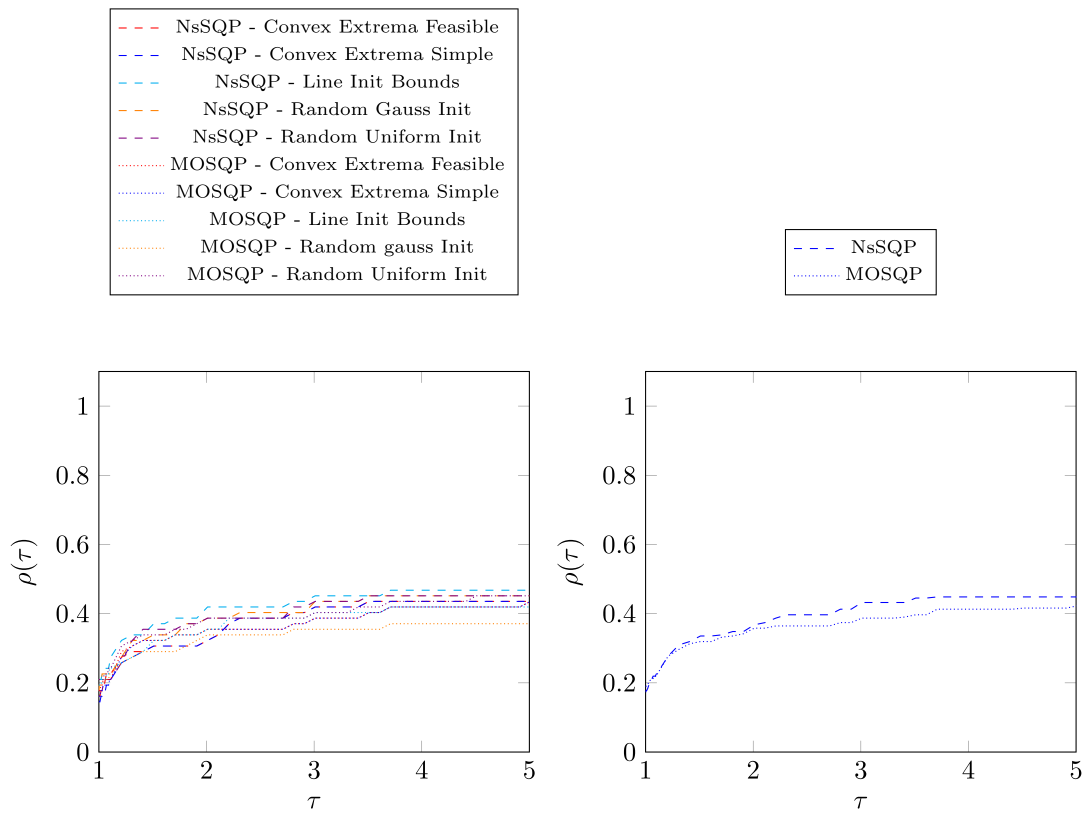
(a) Individual solver performances (b) Averages of the solver

Figure 4.8: Performance profile for MOSQP and NsSQP for the Γ max gap - spread metric.

The performance profiles for the second spread metric Γ again support the overall picture but vary in the details. Figure 4.8a displays the graphs for the performance metric Γ, which measures the maximal gap between samples in the different approximations. This is mainly evident because all graphs of the NsSQP algorithm are above the ones from the MOSQP algorithm. As in the purity metric profiles, the relative order between the different initialisation strategies is preserved, but this is not so pronounced due to a slight variance between the strategies. The total variance is also the most significant difference to the purity metric. Thus, while the difference between the NsSQP and MOSQP algorithms is clear and underlined by the average graphs in Figure 4.8b, the absolute differences between the worst and best overall strategy are much minor.

In addition to this overall comparison, there is one detail that is more apparent in Figure 4.8a than in the previous profiles. This is because the graphs of the MOSQP algorithm in the range $\tau < 2$ can be clearly distinguished into two groups. The first group, represented by the blue and red graphs, shows the curves for the initialisation strategies, including the calculation of the individual extrema.
The graphs of this group run clearly above the graphs of the remaining initialisation strategies, which form the second group. This shows that the inclusion of individual minima is a significant improvement for the initialisation strategies. The reason for the fact that it shows up in

the Γ metric is that the individual minima of all solvers are included in the gap calculation in (4.45). However, the profiles regarding the purity metric in Figure 4.6a also show these two groups within the MOSQP profiles. The gap between both groups is even more significant in the purity, though the graph for "Random Uniform Init" runs between both groups. Why this effect vanishes in the Γ metric when switching from MOSQP to NsSQP is not clear, especially since this switch amplifies it regarding the purity metric. Nevertheless, it can be stated once again at this point that the change to nondominated sorting is an improvement for all strategies.



(a) Individual solver performances

(b) Averages of the solver

Figure 4.9: Performance profile for MOSQP and NsSQP for the hypervolume metric.

The results show that the NsSQP algorithm achieves a better distribution and can compute more nondominated points. As a consequence, it can be expected that hypervolume is also significantly better. However, this presumption is not confirmed when looking at the respective results. Figures 4.9a and 4.9b show the performance profiles for the hypervolume metric. It can be seen that there exists no real difference between both algorithms or the different initialisation strategies. Therefore it cannot be said that the NsSQP algorithm performs better in terms of hypervolume than the MOSQP algorithm, but not worse either. Both algorithms, however, only reach 0.5 as their asymptotic value. This is because, for approximations with only one sample, the hypervolume is 0 and the value of $t^{[p,s]}$ is set to ∞. Thus, the more performance ratios equal to ∞ exist, the lower the asymptotic value. A value of 0.5 shows that the number of problems in the test set to which this applies is not negligible. However, whether this is due to the problems' structure or the algorithm used becomes clear only by comparing it with other algorithms. Therefore we will refer to this comparison later on.

All in all, the used performance metrics show a significant improvement of the algorithm when nondominated sorting is used. This generally confirms the underlying assumption that the cleanup strategy rejects good points in the MOSQP Algorithms, at the same time, it is also shown that the initialisation strategy used can have an equally significant influence. There is a slight tendency that the strategies that take into account the individual minima perform better than the other strategies. However, for the actual application of the algorithm, it should still be considered to combine the initialisation strategies.

Before discussing the data profiles, a closer look will be taken at the differences between the algorithms examined. For this purpose, the metrics' values are first compared without accumulating them as performance profiles. Subsequently, the approximations of both algorithms for selected initialisation strategies and problems are shown.
For a detailed look at the differences, the metric values of both solvers will be compared for each problem individually. For each initialisation strategy, the values of the NsSQP and MOSQP algorithms are compared. The following formula is used for the comparison of the algorithms $s^{[1]}$ and $s^{[2]}$ regarding problem $p \in \mathbf{P}$ to be able to put the differences regarding the individual problems into a better relation to each other.

$$\iota^{[s^{[1]},s^{[2]}]}(p) := \begin{cases} 0, & t^{[p,s^{[1]}]} = \infty \text{ and } t^{[p,s^{[2]}]} = \infty, \\ 1.0, & t^{[p,s^{[1]}]} = \infty \text{ and } t^{[p,s^{[2]}]} \neq \infty, \\ -1.0, & t^{[p,s^{[1]}]} \neq \infty \text{ and } t^{[p,s^{[2]}]} = \infty, \\ \frac{t^{[p,s^{[1]}]} - t^{[p,s^{[2]}]}}{\max_{\bar{s} \in \{s^{[1]},s^{[2]}\}}\left(t^{[p,\bar{s}]}\right)}, & \text{otherwise.} \end{cases} \tag{4.53}$$

Since $t^{p,s} > 0$ it is that $\iota^{[s^{[1]},s^{[2]}]} \in [-1, 1]$.
Figures 4.10 to 4.13 show the values for ι when it is applied to pairs of NsSQP and MOSQP where both use the same initialisation strategy. In all figures the setting is $s^{[1]}$ = NsSQP and $s^{[2]}$ = MOSQP, thus negative values mean the NsSQP metric values are lower and therefore that the NsSQP performs better.
The results so far have supported the thesis that the NsSQP algorithm generally performs better or equal to the MOSQP algorithm. This would mean for all problems in Figures 4.10 to 4.13 that $\iota \leq 0$. However, depending on the metrics, problems with positive ι values can be seen. This fact illustrates the statistical nature of the performance profiles. It is clear that despite the generally better performance of the NsSQP algorithm, there are problems for which the MOSQP algorithm performs better than the NsSQP algorithm.

A look at the differences between the individual metrics confirms the general statement of the performance profiles. The purity and Γ metrics have significantly more values in the negative range, while the Δ and hypervolume metrics are less pronounced. However, since the performance profiles are a quasi accumulation of these values, this is not surprising. However, it is interesting to note that for all metrics, most values are approximately 0, which means that both algorithms perform equally well. This shows that different cleanup strategies do not make any difference for many problems. It is also noticeable that concerning the delta metric, the differences are mainly due to the fact that many values are -1, i.e. the value for NsSQP is not infinite

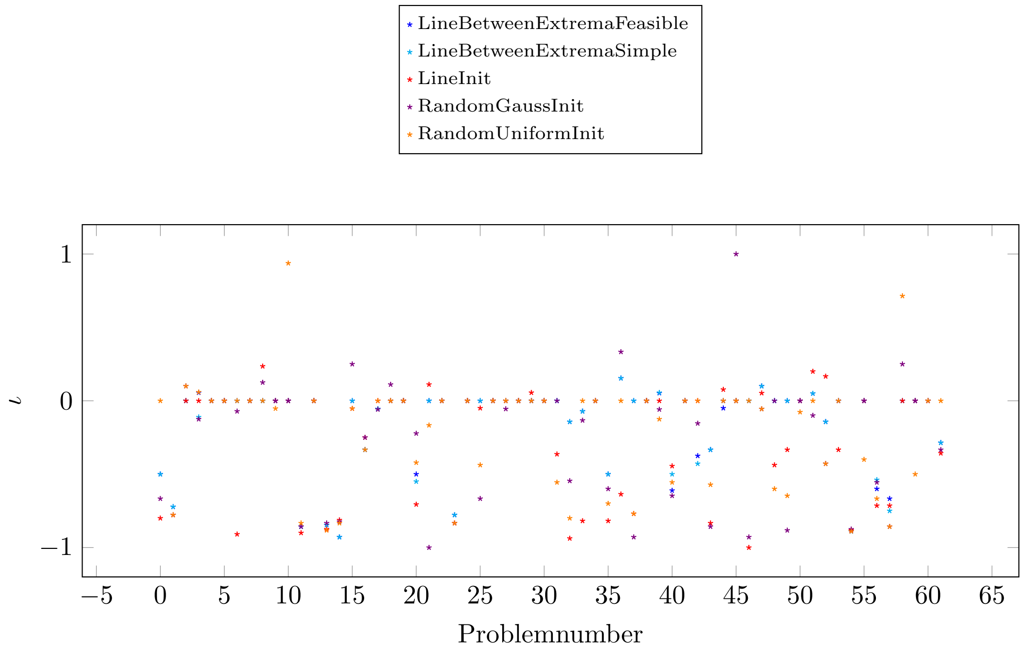


Figure 4.10: Relative differences between the NsSQP and MOSQP algorithm ($\iota^{[\text{NsSQP,MOSQP}]}$, (4.53)) of purity metric (4.42) values for each problem in Table 4.1.

and the value for MOSQP is infinite. This explains very well the fact that the performance profile graphs of the MOSQP algorithm in Figure 4.7a have a lower asymptotic value.

For the hypervolume metric the values are divided roughly equally between positive and negative numbers. This is expected, since the performance profiles are also roughly the same for both solvers.

Overall, the comparison of both algorithms, broken down by problem, offers some additional details, but overall the conclusions of the performance profiles are verified. Finally, it will be looked at how the different metric values come about for individual problems. This will help to examine whether the benefits achieved are explained by the predicted structural improvements through the cleanup strategy. In the following, two approximations are discussed: where the NsSQP algorithm achieves a better value, and two where the MOSQP algorithm performs better.

The improvement of the NsSQP algorithm compared to the MOSQP algorithm is examined here through two problems where the approximations of the NsSQP algorithm achieve a better value of the purity metric. For this purpose the approximations of the NsSQP and the MOSQP algorithm for the “LineInit” initialisation strategy are shown in the Figures 4.14a and 4.14b. Especially Figure 4.14b shows the predicted behaviour, that in cases where the Pareto front is concave, the introduction of nondominated sorting instead of strict Pareto sorting will prevent the algorithm from eliminating points with high potential. In the shown approximations, the NsSQP algorithm is able to compute more points and gains a very good distribution. In this case, the NsSQP algorithm is also able to find a broader Pareto front. This is probably because

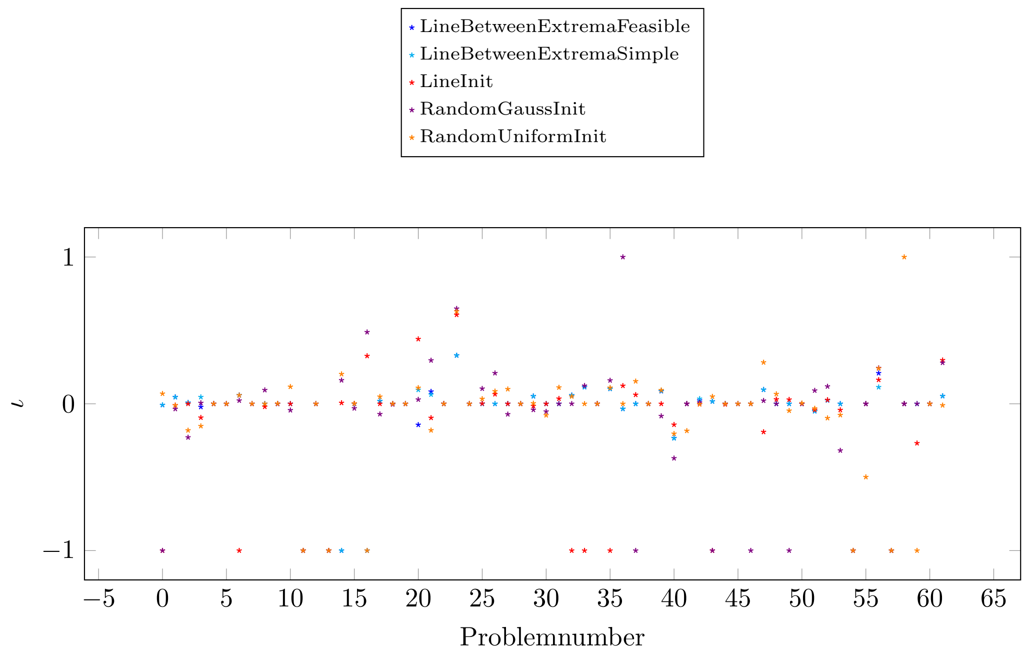


Figure 4.11: Relative differences between the NsSQP and MOSQP algorithm ($\iota^{[\text{NsSQP,MOSQP}]}$, (4.53)) of spread deviation metric (4.46) values for each problem in Table 4.1.

the top left edge of the Pareto front is very flat such that there is a high possibility that points converging to this side are dominated at some intermediate point during the iterations.
Figure 4.14a shows an even more drastic difference between the NsSQP and the MOSQP algorithm. Here the MOSQP algorithm computes only one point, which seems to be one of the individual extrema. In contrast, the NsSQP algorithm seems to find both extrema and can compute more points in between them even though they are a bit clumped. On one side, this shows again the ability of the NsSQP algorithm to compute more points. On the other side, it demonstrates clearly why the purity metric itself is not completely sufficient to evaluate the performance of a solver, since the NsSQP clearly computes more points, but most of them do not add much information to the approximation.
In the worst case, the calculation of these points considerably extends the algorithm's runtime without creating any significant added value. Nevertheless, both examples show that the better values of the NsSQP algorithm in the purity metric are achieved by structural improvements of the algorithm and are not just statistical noise. They reflect precisely the predicted behaviour.

As the figures comparing individual metric values have shown, there are also problems where the approximation of the NsSQP algorithm is worse than the approximation of the MOSQP algorithm. For two of these problems the approximations of both algorithms are shown in Figures 4.15a and 4.15b. As before in Figures 4.14a and 4.14b, the approximations which were computed with the initialisation strategy “LineInit” are shown here. For both problems, the MOSQP algorithm achieves a better value in the Δ spread metric. Thus it achieves a more even distribution on the Pareto front.
In Figure 4.15b it becomes quite clear how this difference comes about. The approximation of

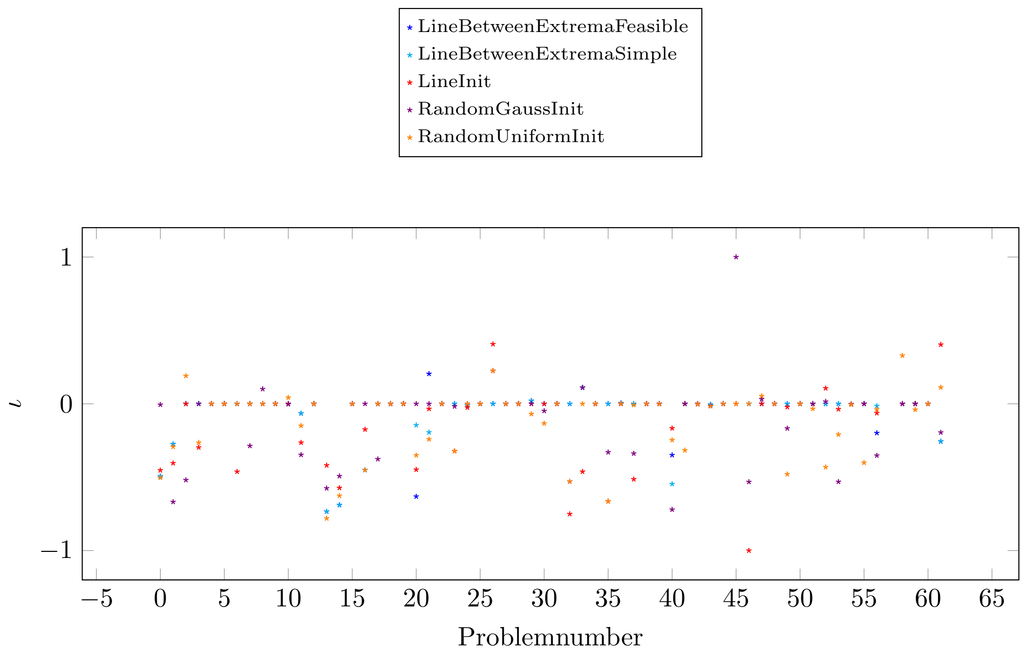


Figure 4.12: Relative differences between the NsSQP and MOSQP algorithm ($\iota^{[\text{NsSQP,MOSQP}]}$, (4.53)) of spread max gap metric (4.45) values for each problem in Table 4.1.

the NsSQP algorithm shows a big gap between two neighbouring samples in the most curved region of the Pareto front. In contrast, the MOSQP algorithm shows a more evenly distributed approximation in this area. Even though both approximations have most of their samples in the lower right part, the NsSQP has more points in a small area. Together with the big gap in the middle, this leads to a worse value for the Δ spread metric. It shows again that many points are computed around the individual minima, which do not add any significant information.
The differences of the approximations in Figure 4.15a are not so clear on first sight. Both algorithms find points in the middle part of the Pareto front. It seems as if the NsSQP has a better distribution in this area. However, a look at the area around the right extreme value shows again that the NsSQP algorithm has computed more points here with a very short distance to each other. Thus, the variance of the gaps between samples is higher, resulting in a worse Δ metric value, which again would lead to a waste of computational power since most of the points do not add value to the approximation. However, the overall result is not as clear as in the other example because the NsSQP also has more points in the middle region. It also finds one more extreme point on the top left side, but if this point really adds to the Pareto front, or is a weakly dominated point is debatable.
Thus, switching from one algorithm to the other would mean a tradeoff between overall approximation and computation time. It seems as if more computed points are not always better since some points do not add value. This might be a structural drawback of the NsSQP algorithm for some problems. However, besides computation time, both algorithms found relatively similar approximations for both problems.
Next, a look at the data profiles will focus more on the computational effort needed in both algorithms.

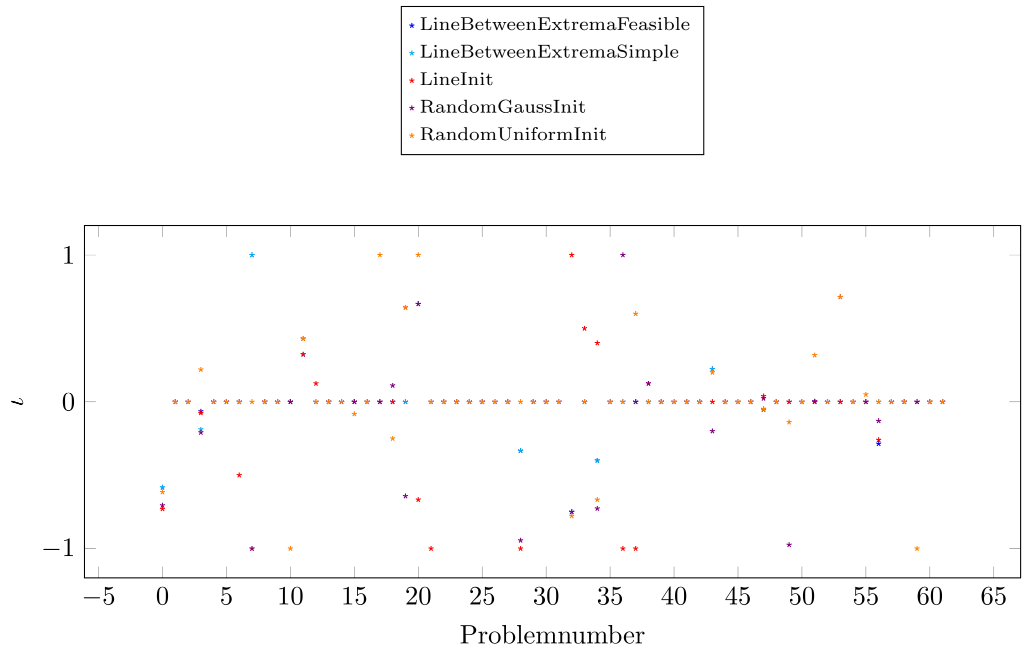


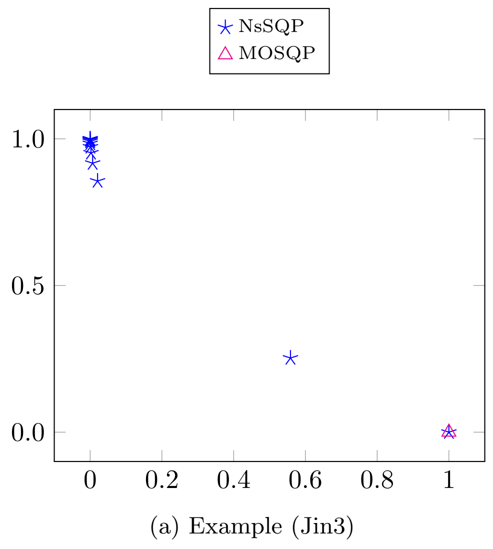
Figure 4.13: Relative differences between the NsSQP and MOSQP algorithm ($\iota^{[\text{NsSQP,MOSQP}]}$, (4.53)) of hypervolume metric (see Section 4.6.2) values for each problem in Table 4.1.

4.7.2 Comparison of Data Profiles for the MOSQP and NsSQP Algorithm

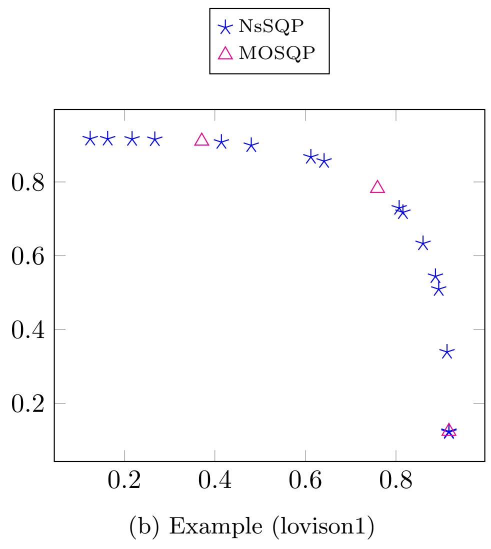
First, the results for the original method of the data profile from [46] are discussed. Figures 4.16a and 4.16b show the graphs for this profile. The left figure shows the graphs for each algorithm and initialisation strategy in the performance profile plots. The algorithms are differentiated by a dashed and dotted line and the initialisation strategies by colour. To get an overview of the differences between the algorithms for the same initialisation strategy, the same strategy has the same colour for both algorithms.

The average graphs in Figure 4.16b show a better result for the NsSQP solver due to the fact that the graph is higher for higher values of σ. However, a look at Figure 4.16a reveals that a single result probably causes this for the strategy “Line Init Bounds” (cyan), which is a clear outlier. The rest of the graphs do not show considerable differences between both algorithms and single strategies. Nevertheless, the NsSQP algorithm with the initialisation strategies based on extreme values shows the best results. Combined with the fact that these are also the variants that achieve the best values for the purity metric, this contradicts the hypothesis that more points lead to a higher computational effort. However, as discussed in Section 4.6.3 the notion of “solved” within the data profiles is biased towards solvers with a better performance in the purity metric. This led to the reformulation of Equation (4.49) into (4.50).

Figures 4.17a and 4.17b show the results for data profiles based on (4.50) as a measurement



(a) Example (Jin3)

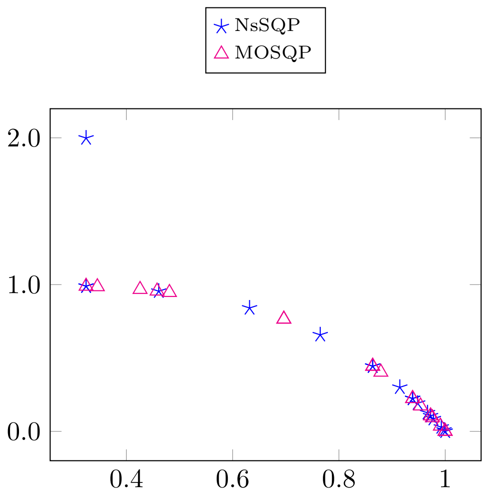


(b) Example (lovison1)

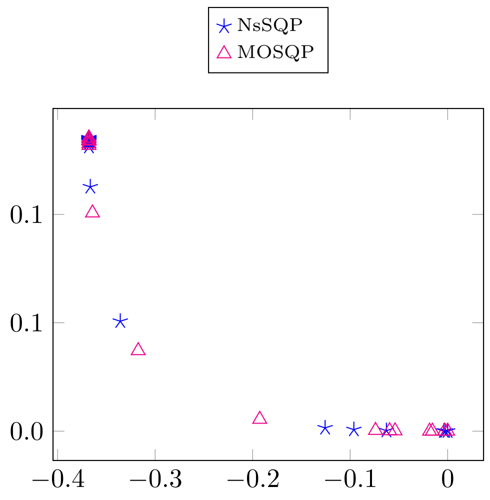
Figure 4.14: Comparison of Pareto front approximations between NsSQP algorithm and MOSQP algorithm for (Jin3) and (lovison1) example.

for “solved” problems. When looking at Figure 4.17b it becomes clear that, on average, the MOSQP algorithm performs better with this formulation of the data profiles. The graph of the MOSQP is clearly above that of the NsSQP for the full profile. A look at the data profiles for each method in Figure 4.17b also shows a higher variance between the methods. Interestingly, the “MOSQP - Line Init between Bounds” variant, which was the negative outlier in the graphs for the original data profile, achieves one of the best results for the modified data profile. In general, the MOSQP algorithm shows better results for each initialisation strategy. However, there does not seem to be a direct connection between the Data Profiles in Figure 4.17a and the results for the purity metric in Figure 4.6a. The graphs for the NsSQP algorithm have a significantly higher variance among themselves than those of the MOSQP algorithm. For the purity metric, this was reversed. Furthermore, the strategies based on individual extrema still achieve some of the best results, even though they also had a high purity metric value. All in all, the hypothesis that a high number of finally computed points leads to high computational effort does not hold when these results are taken into account.

Another statement can be made if the individual strategies are compared with each other. Thus the strategies based on randomness (orange and violet) achieve significantly worse values than the rest, especially when compared with the NsSQP algorithm. This is true for both variants of the data profile but more evident in the modified one. One possible explanation for this is that the randomly initialised points create much waste at the beginning of the iterations, and the actual value within the spread is achieved in later iterations. The points generated on a line between the individual extrema are probably also much closer to the Pareto front than random points in the feasible set. This is especially likely when there are no box constraints or the existing ones are very broad. And since the NsSQP algorithm keeps more of the unpromising randomly generated points in early iterations, this difference is higher in the NsSQP algorithm. Finally, it is also worth mentioning that the variant “NsSQP - Convex Extrema Simple” performs almost as well as the best MOSQP variants. It is also interesting that the strategy “Convex Ex-



(a) Example (Deb513)



(b) Example (T6)

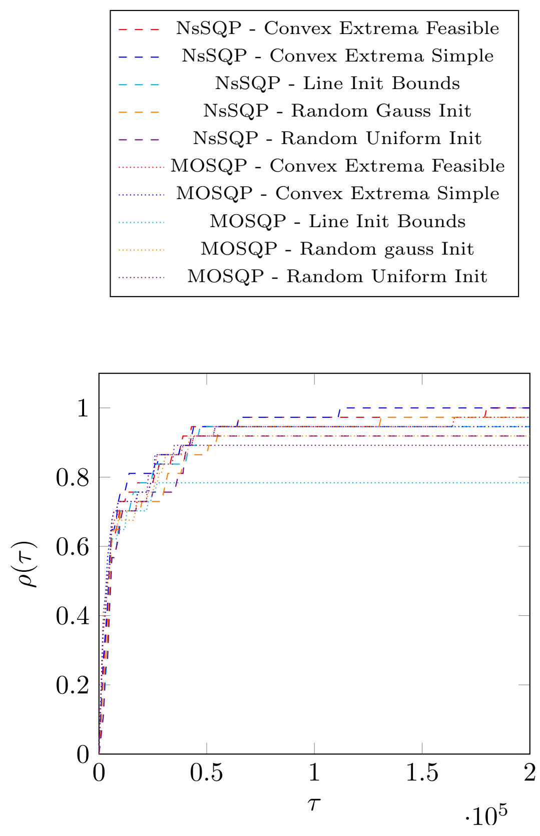
Figure 4.15: Comparison of Pareto front approximations between NsSQP algorithm and MOSQP algorithm for (Deb513) and (T6) examples.

trema Feasible" performs significantly worse than "Convex Extrema Simple" for both algorithms. The function evaluations which are needed within the initialisation strategy are counted as well for the profile. Thus the better performance of the simple version gives a hint that the feasibility correction in the initialisation strategy does not help much with regards to convergence and thus wastes computational power.

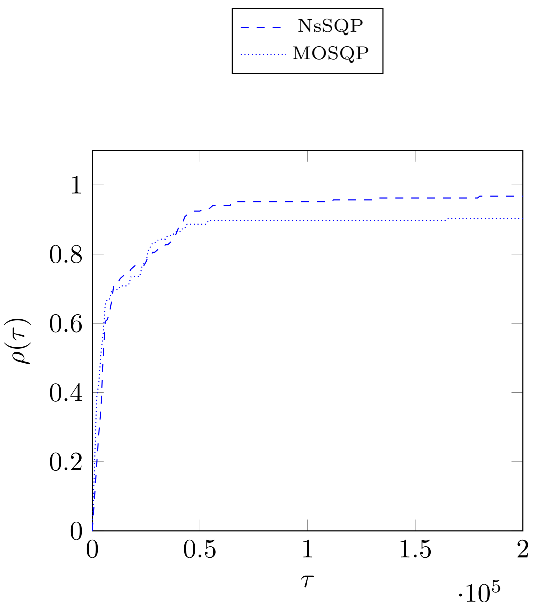
Overall, it can be seen that the NsSQP algorithm requires a slightly higher computational effort. However, this is only valid for one of the data profile variants and is mainly induced by the outliers based on random initialisation.

4.7.3 Conclusion of the Comparison of the MOSQP and NsSQP Algorithm

It was shown that the NsSQP algorithm is able to compute more nondominated points, which results in clearly better purity metric values. This is true for all variants of the initialisation strategies. All variants of the NsSQP are better than the best variant of the MOSQP algorithm. The same holds for the Δ and Σ spread metric, even if the differences are not apparent. Even if the difference in hypervolume performance metrics is not that clear, it is clear that the NsSQP algorithm consistently achieves better or the same results on average over the whole test set for each individual initialisation strategies.
However, by looking at the metric values for each problem individually, it was shown that the improvements are statistical ones, and that there are problems where the MOSQP algorithm performs better on single performance metrics. But it was also shown the the NsSQP algorithm for a problem with a concave Pareto front indeed shows the predicted structural advantage.
A hypothesis was given that the higher number of points the NsSQP algorithm is able to compute, sometimes leads to wasted computational power. However, it was shown that this does not result in worse data profiles.



(a) Individual solver performances



(b) Averages of the solver

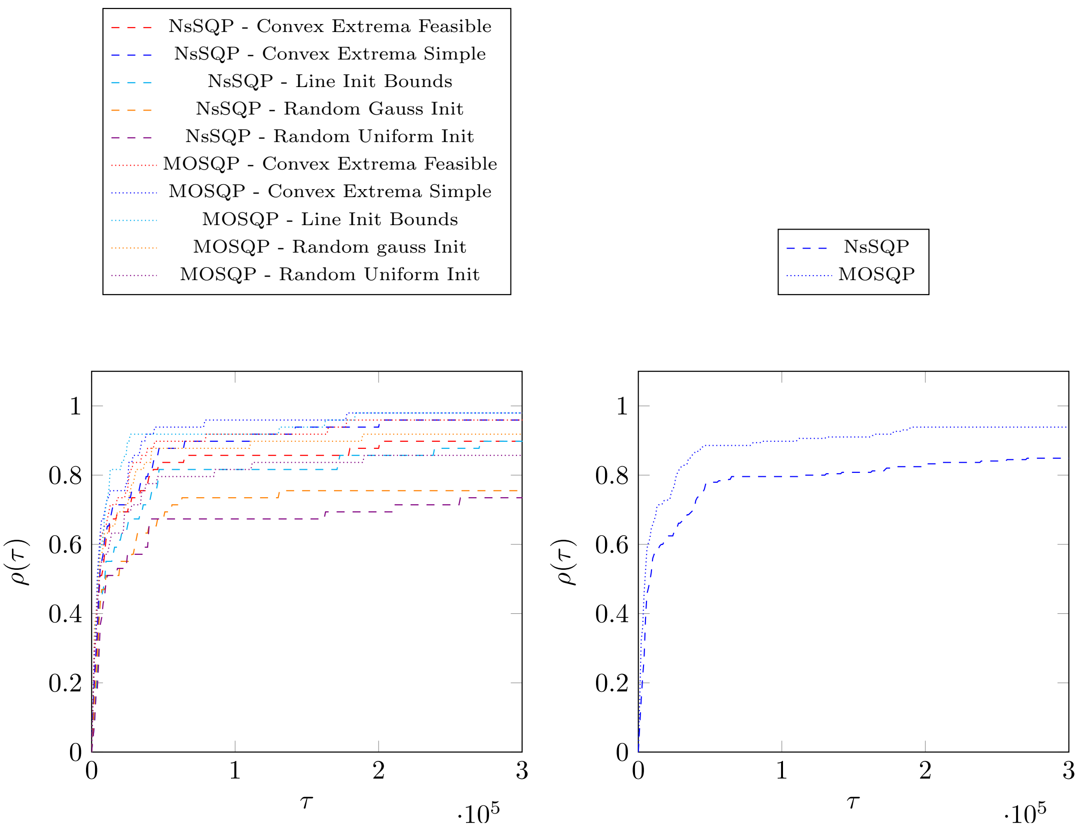
Figure 4.16: Data profile for the NsSQP algorithm and the MOSQP algorithm based on the intersection purity metric (4.49).

An example considered indicates that initialisation strategies based on randomisation perform significantly worse than others. This effect is somewhat amplified in the NsSQP algorithm, which on average, performs worse. For strategies that take into account individual minima, however, the difference is much smaller.

The final result is that the use of nondominated sorting and thus the NsSQP algorithm significantly improves the MOSQP method. However, the initialisation strategies also have a significant effect. Here, the strategies based on individual minima and especially the strategy "Convex Extrema Simple" perform the best.

4.8 Comparison of NsSQP, NSGA-II and Weighted Sum Algorithm

After comparing the NxSQP and MOSQP in the previous section, this section compares other widely used approaches for multi-objective optimisation. As in the original MOSQP paper [46]



(a) Individual solver performances (b) Averages of the solver

Figure 4.17: Data profile for the NsSQP algorithm and the MOSQP algorithm based on the intersection percentage metric (4.50).

the NSGA-II from [27] is used for comparison as one of the most common heuristical approaches. Additionally, the weighted sum method is used to represent avery simple but widespread scalarisation approach.
The results in the paper show good performance of the MOSQP algorithm and the weighted sum method in purity, spread (4.46) and data profiles but better performance of the NSGA-II with regard to the spread metric (4.45). This Section will do a similar comparison of the three methods. However, it should be noted that even though the same methods are used for comparison, all three are different implementations than the ones in the original MOSQP paper. Also a test set with many more problems and different problem types is used. Thus the results may differ even in areas not corresponding to the modification of the MOSQP algorithm itself.

4.8.1 Comparison of Performance Profiles for the NsSQP, NSGA-II and Weighted Sum Algorithm

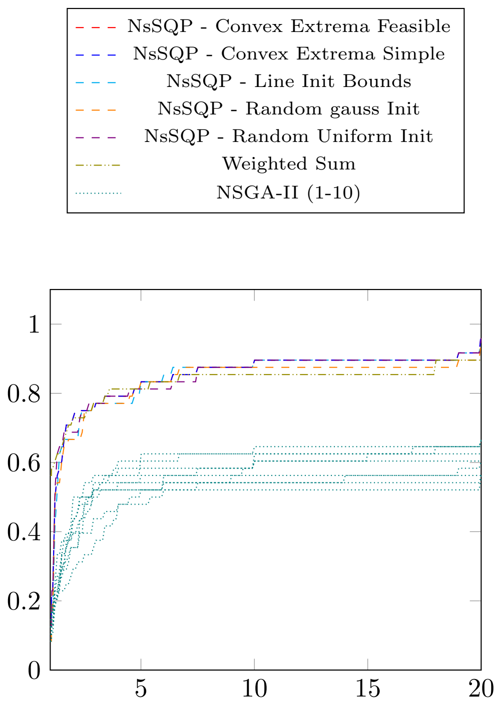
As in Section 4.7, the algorithms are first compared with profiles build on the performance metrics for purity, spread (deviation and max gap) and hypervolume. For the NsSQP algorithm, the same versions and settings are taken into account as in Section 4.7.

For the comparison with the NSGA-II algorithm, the C++ implementation in the Pagmo-2 solver [9] is used and interfaced such that it uses the same problem classes as the other algorithms. However, since the NSGA-II algorithm is based on random processes, it is necessary to show the results of multiple runs. In contrast to [46], all runs are taken into account and shown instead of only the best run. This helps to get an overview of how good the NSGA-II algorithm can be at its best and it shows the variance between different runs.
It is important to note that the implementation used in the of the NSGA-II algorithm is not able to handle nonlinear constraints. Thus, the test set is restricted to unconstrained and box-constrained problems. The comparison is still considered fair since the handling of nonlinear constraints is in general considered to be one of the advantages of derivative-based algorithms.
The NsSQP algorithm as well as the weighted sum method are implemented using the C++ interface of WORHP. For the generation of weights, the same strategy as in the convex initialisation between extrema Algorithm 4.23 is used. This means that the weights are evenly spread, but the number of points is not always precisely the same as the maximal number of points allowed in the other algorithms.

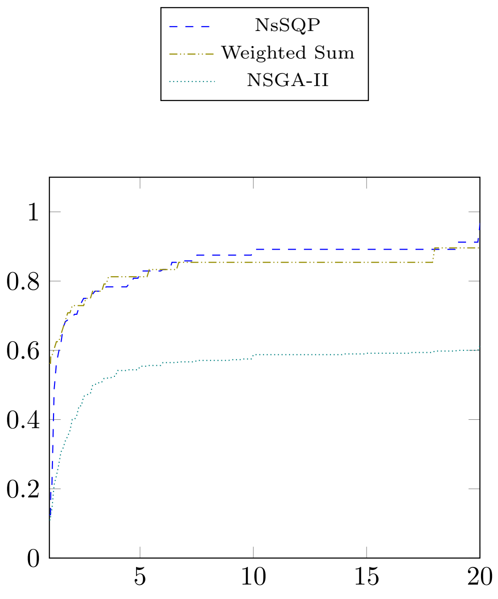
As in Section 4.7.1, the NsSQP algorithm has 20 iterations in the spread phase and 20 in the refinement phase. Thus, the weighted sum method and the NSGA-II algorithm both are set to 40 iterations. All three algorithms work with a maximal number of 20 points.

Note 4.28
In general it is not specified how the single-objective problems formulated by the weighted sum method are solved. It is possible to solve them using an SQP solver as well as derivative free heuristic methods. However, usually they are solved using derivative based methods. In addition, the implementation used in this thesis uses the SQP method in WORHP. Thus, the weighted sum method will be considered a derivative based method for the remainder of this Chapter.

First the purity metric is evaluated for all algorithms. Figures 4.18a and 4.18b show the resulting profiles. As in Section 4.7.1 Figure 4.18a on the left shows the profiles and Figure 4.18b the averages of the solvers. Since the weighted sum method has only one variant and run, the graphs in Figures 4.18a and 4.18b are the same.
In Figure 4.18a two things are immediately comprehensible. First, the different runs of the NSGA-II algorithm have a high variance with respect to the purity metric. The variance is much higher than the one for all NsSQP variants and the weighted sum combined. This clearly shows the random behaviour of the NSGA-II algorithm.
Second, there is a clear gap between all runs of the NSGA-II and the derivative based methods weighted sum and NsSQP. The cause for this is with high probability the property of a proven good convergence towards Pareto-optimal points. It is also not very surprising since this very property is one of the well-known advantages of derivative based methods compared with heuristic methods [14, p. 3]. However, some observations raise questions related to this argument. First, the results for the purity metric when only the NsSQP and MOSQP algorithms are compared in Figure 4.6a show a high variance. Second, there is a visible difference between the profile of the weighted sum method and the average of all profiles of the NsSQP in Figure 4.18b. The first point is explained by the fact that performance profiles do not stay the same for one algorithm when another algorithm is added. Thus, if only derivative based methods are compared, differences show up which are not visible when compared with completely different approaches like the NSGA-II. The second point is probably explained by the fact that the weighted sum is allowed to compute slightly more than 20 points if the algorithm for the weights cannot compute exactly 20 evenly spread points. See algorithm 4.23 in Section 4.5.7 for more details. One could



(a) Individual solver performance



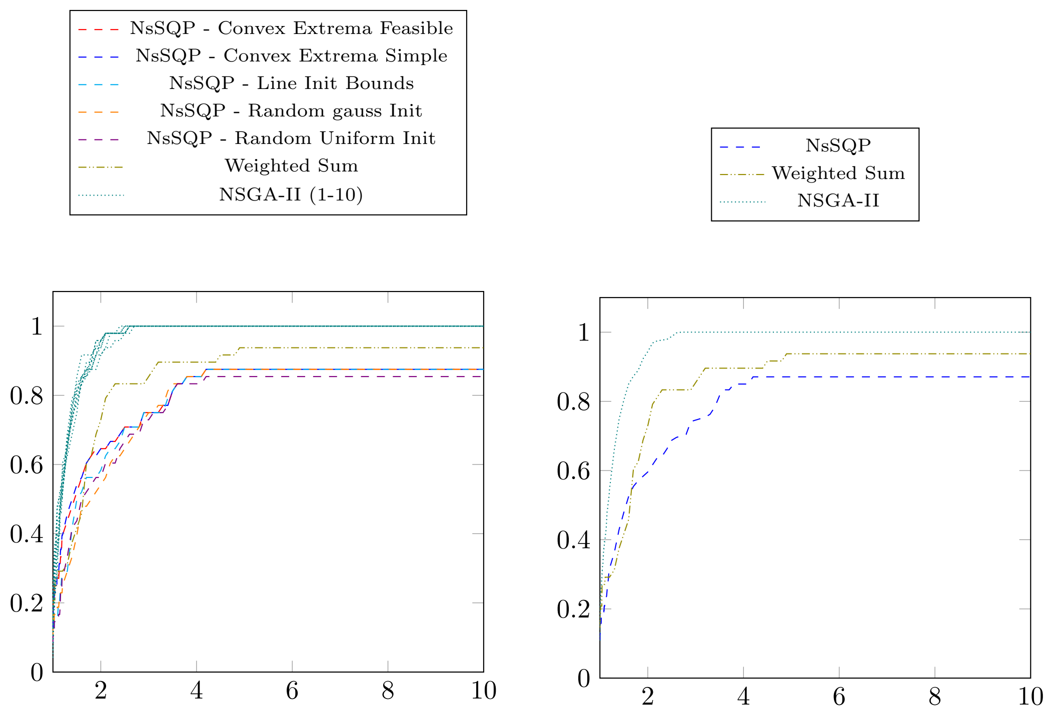
(b) Averages of the solver

Figure 4.18: Performance profile for NsSQP, NSGA-II and weighted sum for the purity metric (4.42).

argue that, for better comparison, some points computed by the weighted sum should be deleted to get exactly 20 points. However, this would strongly influence the spread and hypervolume metrics and thus would not lead to a better comparison.

Figures 4.19a – 4.20b show the results for both spread metrics. Since they show mostly the same behaviour, they are discussed together. Both, the δ (4.46) and the Γ (4.45) profiles show a clear advantage of the NSGA-II algorithm over the SQP based methods. Again, this is not surprising because the good spread of the NSGA-II algorithm is its well-known advantage, especially since the test set includes problems with unconnected Pareto fronts and problems with several local minima. Especially the local minima explain the gap between the NSGA-II and the SQP based methods regarding the max gap Γ metric in Figure 4.20a. Since the metric considers the gaps in between points of one solution and the gap to the best-known values for individual minima, it results in bad values for algorithms when the algorithms were not able to compute the global optimum for an individual objective function. Thus, the local characteristic of SQP methods do not perform well against the global NSGA-II solver [14, p. 3].

Interesting is the difference between the weighted sum and the NsSQP method. Figure 4.19a shows that the the weighted sum performs better than the NsSQP with respect to the δ metric but worse with respect to the Γ metric shown in Figure 4.20a. In short, this result shows that the weighted sum method computes non evenly distributed points which are still in regular dis-

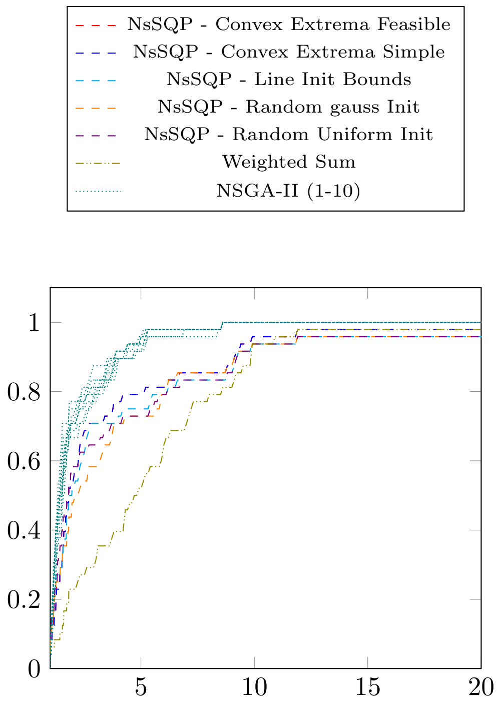


(a) Individual solver performance

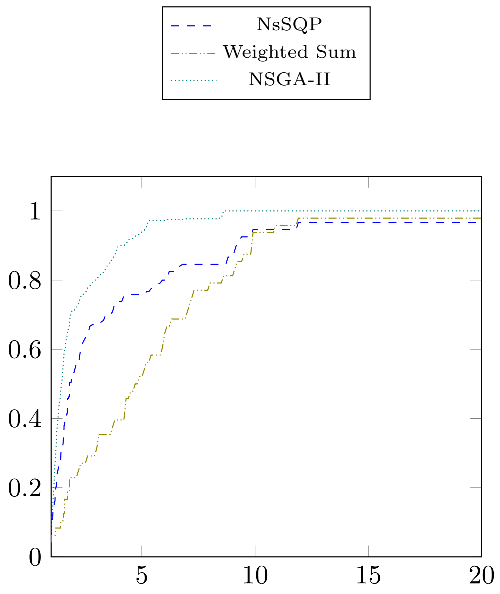
(b) Averages of the solver

Figure 4.19: Performance profile for NsSQP, NSGA-II and weighted sum for the Δ (deviation -) spread metric (4.46).

tribution. Again, the first point is well known and one of the main criticisms of the weighted sum Algorithm (see [25]). It also leads to bad values in the max gap Γ metric because usually the weighted sum computes strongly clustered solutions. The fact that the NsSQP algorithm performs much better in the Γ metric shows that the scalarisation used in the refine phase (4.33) and especially the additional constraints in it prevent the algorithm from suffering from the same problem even though the used scalarisations can be understood as weighted sum approaches.
Despite the well-known fact that the weighted sum generally does not compute a well spread Pareto front, the profile of the Δ metric shows a good performance, which is better than the one of the NsSQP and not much worse than the NSGA-II. Maybe the most important lesson from this is that considering only one of the profiles alone does not lead to a well-differentiated statement about the algorithms. Since clusters of solutions can be well spread when compared only to themselves, it becomes clear why the weighted sum reaches such good values, even if the individual minima of all solvers are considered, because some outliers usually do not have a high effect. However, the results in Figures 4.19a and 4.19b should not be neglected completely based on these arguments. Experience shows that while the weighted sum method computes clustered solutions, the solutions are also spread over the whole front between the computed minima. Often a monotonically increasing difference between points can be observed when going from one individual extremum to another (see example in [25]). Thus the distribution is somehow regular, especially in comparison with the Pareto fronts computed by the NsSQP where the size of gaps between points does not follow a certain pattern (see for example Figure 4.15b).



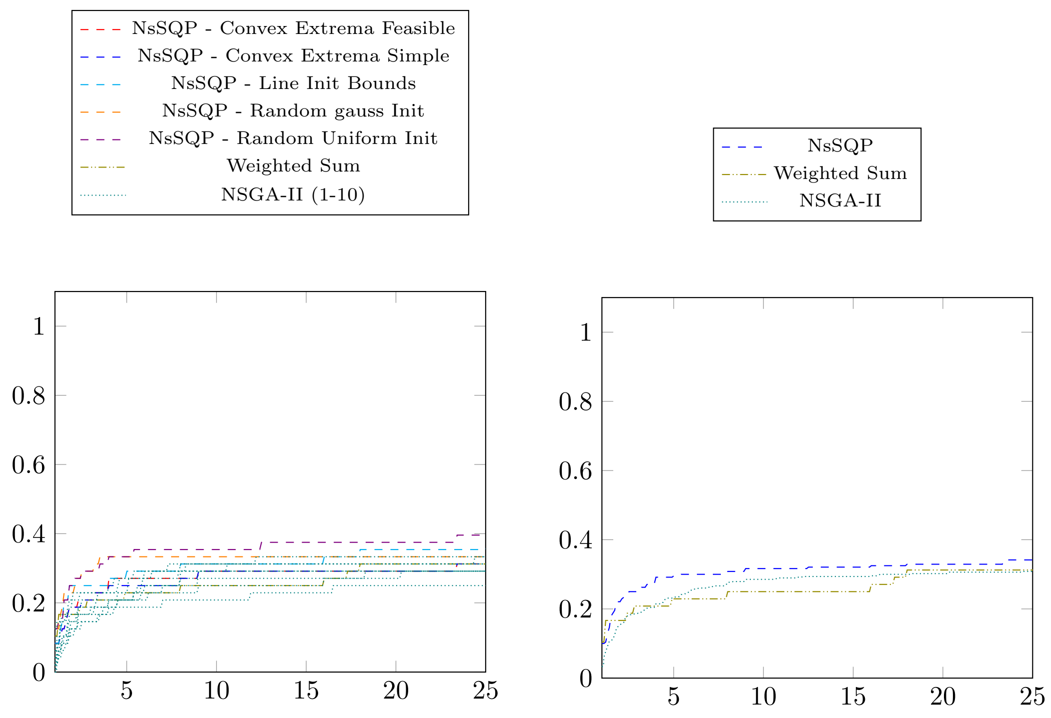
(a) Individual solver performance



(b) Averages of the solver

Figure 4.20: Performance profile for NsSQP, NSGA-II and weighted sum for the Γ (max gap -)spread metric (4.45).

Last, the results regarding the hypervolume metric are shown. As the hypervolume incorporates both spread and purity, it is interesting to investigate how the clear advantage of derivative based methods regarding purity and the advantage of the NSGA-II regarding spread will sum up. Figures 4.21a and 4.21b show the resulting profiles of all algorithms. The first thing to notice is that as in the comparison of NsSQP and MOSQP in Figure 4.9a all profiles in Figure 4.21a stabilise below the value of 0.5, which is because solutions with only one point have a hypervolume of 0, leading to an infinite value for the corresponding performance metric value. Interestingly, the NSGA-II stabilises on the same level as the derivative based methods, showing that the global characteristic of the NSGA-II algorithm does not lead to less solutions with only one point.
A look at the averages of the three algorithms in Figure 4.21b shows that regarding the hypervolume, the performance of all three algorithms is almost the same with a slightly better performance of the derivative based methods and especially the NsSQP algorithm. However, the variance between different runs displayed in Figure 4.21a shows that this slight advantage might disappear when another set of runs is used for the NSGA-II algorithm. Therefore, it can not be said that the derivative based methods have a structural advantage in the algorithm.



(a) Individual solver performance

(b) Averages of the solver

Figure 4.21: Performance profile for NsSQP, NSGA-II and weighted sum for the hypervolume metric.

4.8.2 Comparison of Data Profiles for the NsSQP, NSGA-II and Weighted Sum Algorithm

For the comparison regarding data profiles, all four variants for data profiles described in section 4.6.3 are used. Since one of the main ideas behind the modified data profiles was to design them a bit fairer for randomly based heuristics like the NSGA-II, it is interesting to see which difference they make.
Figures 4.22a - 4.22d show the results regarding all four metrics. It is directly obvious that the different metrics do not make a big difference regarding the gap between the NSGA-II and the derivative-based methods. They do, however, make a difference for the single initialisation strategies of the NsSQP algorithm. Especially the two methods using randomly initialised points perform differently on each metric. That is why it should be remembered that even though derivative-based algorithms do have a proven convergence towards the Pareto front, the result for two different runs with randomly chosen initial points may still differ. Therefore is possible that a solution of the algorithm is not considered solved in terms of data profiles (see Section 4.6.3) even though it converged. For the derivative-based algorithms, this is supported by the fact that two variants with random initialisation perform equally good as the rest when the notion of solved is defined by dominance in Figure 4.22d.
However, this difference between each data metric does not show up significantly when considering the NSGA-II algorithm. Not even the profile which considers undominated points as solved

shows a notable number of solved problems by the NSGA-II algorithm. This might be coupled to the bad performance in the purity metric shown in Figure 4.18a.
The comparison of data profiles yields the result that it is hard to measure needed function calls when it is unclear how to define the notion of solved. However, the necessity to find new definitions and the missing canonical definition shows one of the disadvantages of heuristic methods. These problems also hint at general challenges in multi-objective optimisation.

4.8.3 Conclusion of the Comparison of the NsSQP, NSGA-II and Weighted Sum Algorithm

The NsSQP algorithm was compared to the NSGA-II algorithm and the weighted sum method. Both are widespread representatives of certain types of approaches for multi-objective optimisation. Before, Section 4.7 showed that the modification of the MOSQP algorithm yielding the NsSQP algorithm overall improves the performance of the algorithm in almost all aspects. However, the results from this section show that the modification does not engender significant changes with regard to the main structural differences between the approaches. Thus, the NSGA-II still yields a better overall spread but has significant problems in the purity metric and also regarding data profiles.
The fact that implementation from the ESA like the Pagmo implementation does not handle nonlinear constraints at all also shows that problems with a high number of those constraints are considered to be a strength of derivative-based algorithms like the NsSQP, MOSQP and weighted sum algorithms.
For these reasons, it can be said that the NsSQP and MOSQP algorithms represent another useful tool for solving multi-objective optimisation problems in addition to the previously known and widely used approaches. They fill a gap between methods with proven convergence towards the Pareto front which require a-priori known hyperparameter, and algorithms that do not require a-priori knowledge but are heuristic. Thus complementing widespread methods like the weighted sum or the NSGA-II.

4.9 Conclusion and Outlook on the Advances to the MOSQP Algorithm

In this chapter, advancements to the MOSQP algorithm were described. By usage of the nondominated sorting technique, the algorithm could be improved and thus became the NsSQP algorithm.

To achieve this, first, the fundamentals of the MOSQP algorithms were explained in Section 4.1. General properties like the proof of convergence towards the Pareto front were shown. Section 4.2 explained the original idea of cleaning up the current set of points in intermediate iterations. It was shown in which case the purely Pareto based clean-up strategy will lead to problems.
Then this thesis presented the new idea of replacing the Pareto based strategy with the clean-up strategy used by the heuristic NSGA-II algorithm. The latter introduces the idea of several levels of nondominated points, thus leading to a more robust clean-up. It was described how to integrate the new strategy into the existing algorithm.

In Section 4.3 it was described how the MOSQP algorithm is implemented using the SQP solver WORHP (see Section 2.3), thus utilising the advanced features of WORHP for regulari-

sation, step sizes and general build-up of QPs.
Last, this thesis explained different initialisation strategies in Section 4.5. Thereby extending the original MOSQP strategies and generalising them for higher dimensions.

How the advancements of using the nondominated sorting and the different initialisation strategies performed was investigated in Section 4.7. Evaluating multiple performance profiles for the NsSQP and MOSQP algorithm with all initialisation strategies showed that especially the usage of nondominated sorting improves the performance significantly. Also, the dependency of the resulting Pareto fronts on the initialisation strategy used was shown. Here, mainly the strategy using convex combinations between extrema, which was introduced and generalised in this thesis, performed very well in all performance metrics.

Subsequently, in Section 4.8.1, a comparison between the NsSQP algorithm, the weighted sum and the NSGA-II was made based on performance and data profiles. It was shown that the SQP based methods compute a more accurate approximation of the Pareto front, whereas the NSGA-II gets a better spread but usually finds points that are dominated by the ones from the SQP based methods. Also, the NsSQP was able to compute an approximation with smaller gaps than the weighted sum, thus finding a better spread over the whole front.
With data profiles, the general problem of an unclear convergence and a high number of function evaluations in the NSGA-II algorithm was demonstrated.

All in all, it can be said that the MOSQP algorithm, especially with the modification to the NsSQP algorithm, is a high-performance tool for multi-objective optimisation, whose designated areas of application are especially where there is little prior knowledge about the problem structure, and high convergence and accuracy are required.

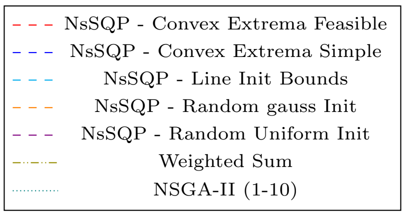
This thesis shows how big the impact of the used clean-up strategy within the MOSQP and NsSQP algorithm is. One idea to find even better strategies than the nondominated sorting could be to use modern machine-learning techniques like reinforcement training combined with neural networks. A similar idea is successfully used in [79] for steering interior-point methods.
Within the MOSQP algorithm, it could be worth trying a neural network to decide which points to take from one iteration to the next. These neural networks might be able to model quality criteria that are harder to define as explicitly. A challenge here would be to define good training problems. Since reinforcement learning needs a high number of repeatedly changing problems, the test-sets used in the literature would be far too small. This problem could be solved by using automatically generated polynomials as constraints, thus creating various shapes of Pareto fronts. However, investigating how good a neural network trained in this way would generalise on other problem classes would then be obligatory.
However, this idea might have the potential to increase the efficiency of the MOSQP algorithm and could later also be used in genetic algorithms like the NSGA-II. A possible disadvantage of this idea is that the behaviour of the algorithm would be harder to predict and understand since it could be hard to understand on which basis the neural network orders the points.

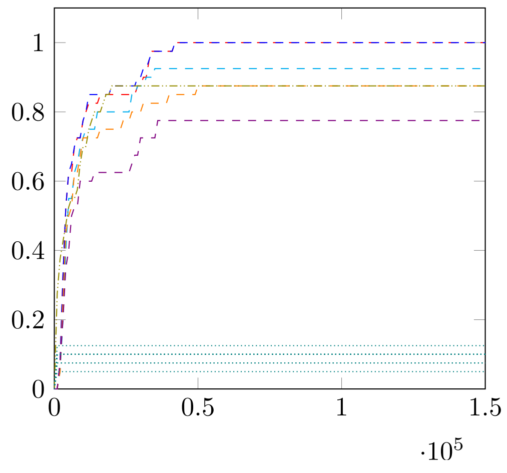
Another idea is to replace the formulation used in the refinement phase (4.33) by a formulation based on the Pascoletti-Serafini scalarisation (see Section 2.5.3) instead of the weighted sum with constraints. In this way, problems with constraints leading to empty feasible sets could be solved, and also the spread obtained in the spread phase might be better preserved. However, since this would change the fundamental structure of the MOSQP algorithm, the convergence proof would need to be adapted to the new formulation.

Similarly, the formulation in the spread phase could be adapted in a way to use different combinations of multiple objectives for the points in each iteration. Thus, more possible directions would be possible for a single step, which could be used to steer the optimisation in areas of the Pareto front where less information is known so far. This steering could be managed by another neural network, similar to the idea described above. However, this idea would also introduce the necessity to adapt the convergence proof and would have the same challenges as the idea for reinforcement learning described above.

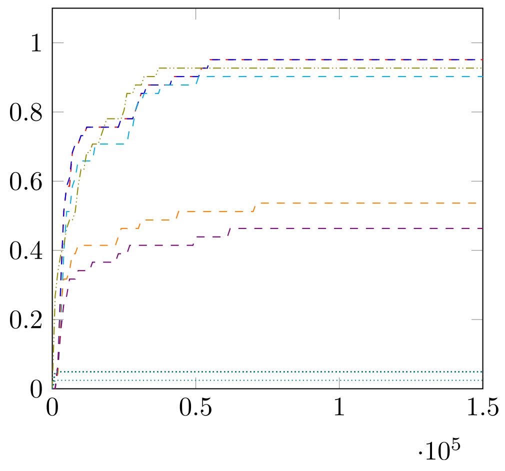
Last but not least, an idea would be to combine the MOSQP and NSGA-II algorithms to utilise the advantages of both. This could be done by simply letting the NSGA-II and MOSQP spreading phase calculate a well-spread approximation which would then be refined with the technique of the MOSQP algorithm, thus getting the precision of SQP based methods. Another more complex approach could be to see a SQP step as another way of mutating a point x within the NSGA-II algorithm. Thus the algorithms are enriched with the ways of finding new points and combined directly with the derivative-free methods of the heuristic. For this, both formulations, the one from the spread phase and the one from the refinement phase, could be used. This idea would be highly experimental and thus would require a detailed examination of the interdependencies between both methods within the iterations. However, on the success, it could be a way to combine both worlds seamlessly, thus leading to an entirely new type of hybrid algorithm.

All in all, there are several promising ideas on how to proceed. However, since they all require fundamental changes within the implementation and utterly new result examinations, they are out of the scope of this thesis.

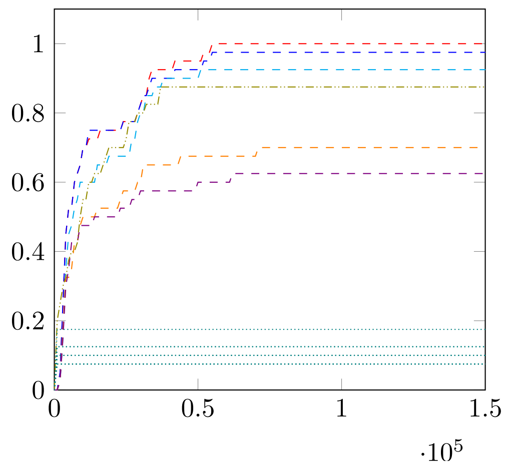

NsSQP - Convex Extrema Feasible
NsSQP - Convex Extrema Simple
NsSQP - Line Init Bounds
NsSQP - Random gauss Init
NsSQP - Random Uniform Init
Weighted Sum
NSGA-II (1-10)



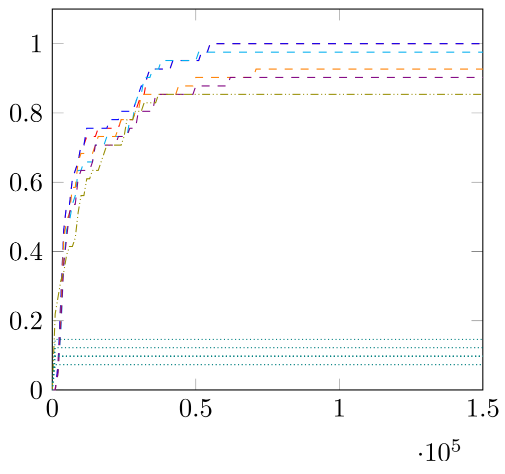
(a) Intersection Purity Metric (4.49).



(b) Intersection Percentage Metric (4.50).



(c) Intersection Percentage Metric (4.50) using Average Gap Intersection (4.51).



(d) Intersection Percentage Metric (4.50) using Dominance Intersection (4.52).

Figure 4.22: Data profiles of NsSQP, NSGA-II and weighted sum.

Chapter 5

Application to a Bi-Objective Trajectory Optimisation Problem from the Context of Autonomous Ship Control

Contents

Ship manoeuvring, especially in close harbour scenarios, is a very complex task. Increasing numbers of ships within a harbour, ranging from big container ships to small one-person sailing ships, participate in the harbour traffic. It is a challenging task to manoeuvre a ship safely on an optimal route through the harbour. The projects GALILEOnautic and GALILEOnautic 2 [52, 84, 107] research how assistant, automation, and autonomous systems (called automated systems in the following) can be implemented for ships.

In the heart of these automated systems is a trajectory optimisation module. The term trajectory refers to a set of waypoints that give certain states and the controls necessary to reach these states over time. To build automated systems in these scenarios, it is necessary to predict how the ship will behave in the future and how controls like the rudder or propeller influence it. When this prediction can be computed, it will then be used to find optimal trajectories for specific criteria.
However, in reality, this criteria represents the combination of several different objectives. These can be criteria like energy consumption or other emissions and criteria to drive the control process, like coming as close as possible to a given final position or reaching a stable point at the end of the current computed trajectory. In this context, stable means that no movement happens anymore. Also, objectives that increase the comfort of the crew and passengers like the radii of curves, acceleration forces, or just control oscillation can be considered. Last, some objectives stabilise the computation of the system numerically. Overall, this demonstrates that, in most cases, a trajectory optimisation problem is a multi-objective optimisation problem.

When implementing a controller for this scenario, one option is to use nonlinear model-predictive control (nMPC) - a numerical method whose main idea (formulated e.g. in Lee and Markus [51]) is to repeatedly solve the original problem on a short rolling horizon and to implement only the first part of the solution to the system [113]. This technique has the advantage that it takes nonlinear dynamics and path constraints into account [97], but on the cost fo being computationally more expensive. For a general overview see the Book by Grüne and Pannek [60]. Beside the usage in the above mentioned projects GALILEOnautic and GALILEOnautic 2 [107] nMPC is for example applied in the automotive sector [28, 86, 101, 106, 122], in space-applications [29, 88, 109] or in chemical reactions [47]. More applications are for example mentioned in [103].
The idea is roughly to compute an optimal trajectory for a specific prediction horizon. Then the first few control signals based on this trajectory are applied. Subsequently, the prediction horizon is moved forward based on the elapsed time, and the subsequent trajectory is computed. This whole procedure can happen at a high frequency like several times in a second. Since trajectory optimisation is central to such an nMPC module, it is essential to understand the behaviour of the solutions in all situations and how the composition of multiple objectives into one single-objective function manipulates them.
Of course, this kind of controller must be able to compute one single trajectory. Thus to compute the whole Pareto front for a given multi-objective optimisation problem is not immediately applicable. The topic of multi-objective nonlinear model predictive control is a current area of research and, for example, described in [65, 100, 113]. However, this thesis does not focus on the extension of multi-objective optimisation to the real-time process of nMPC. Instead, it investigates how the information gained through multi-objective optimisation techniques and especially the addition of parametric sensitivities can guide and accelerate the development process for an nMPC module that might later be implemented as a single-objective nMPC module. In addition, this chapter provides ideas on how to utilise the described methods within an actual multi-objective nMPC module as an outlook at several places.

This chapter aims to investigate if and how the techniques described in this thesis can help during the development process of an nMPC module for autonomous ships.

To reach this goal, first Section 5.1 describes the trajectory optimisation problem for an autonomous ship. Second Section 5.2.1 evaluates the general applicability of the individual approaches and how the changes proposed in Chapter 4 impact the performance of the MOSQP / NsSQP approach. Last, Section 5.2.2 shows which information and enhancements can be gained from the approach based on parametric sensitivity analysis from Chapter 3.

Note 5.1
This chapter aims at discovering the benefits multi-objective optimisation can have on the development process of an nMPC module. The goal is explicitly not to develop the best model and algorithm to implement an nMPC module for autonomous ships but instead to help the developers. This chapter takes a snapshot of the model and objective function under investigation during the development process within GALILEOnautic 2. This note emphasises that some problems in the model and nMPC formulation are deliberately taken into this chapter to demonstrate how the techniques described help discover them. However, even though the model and objective functions do not represent the final formulation used for steering the ships in GALILEOnautic 2, they represent a realistic situation within the project's development process.

Note 5.2
As described in Note 5.1 the goal of this Chapter is investigating multi-objective algorithms for the usage in optimal control. The goal is not, however, to find the best controller for an autonomous ship. For this reason, this thesis does not provide any theory about optimal control. All related problems are investigated directly in a discretised formulation, thus mainly the optimality criteria described in Chapter 2 apply. For details on the theory of optimal control, see the book by Betts [8]. For a description of how to discretise optimal control problems using direct methods, see the description of TransWORHP, the transcription framework based on WORHP in Knauer and Büskens [74].

5.1 Problem Description for an Autonomous Ship Controlled via a POD-Drive

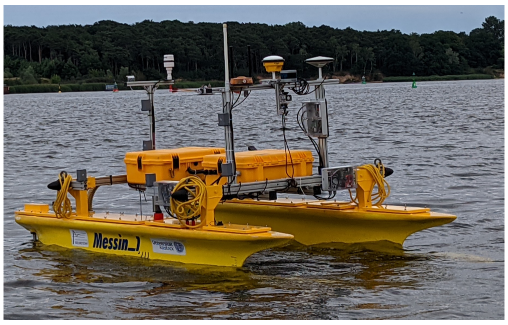


Figure 5.1: Picture of Messin in the harbour of Rostock.

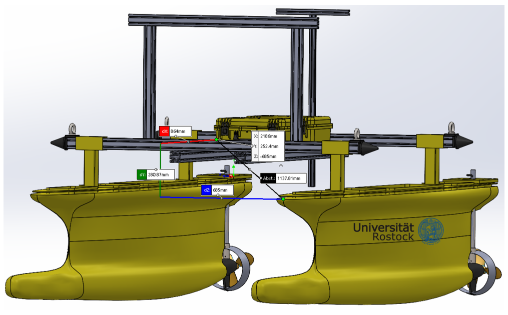


Figure 5.2: 3D-model of Messin provided by [116].

In this thesis, a ship with a so-called POD drive is modelled. The ship has no classical rudder in this type of drive but is propelled and steered with one or more propellers rotating around their vertical axis. The force induced by the propeller can be directed to either accelerate the ship forward or to cause a rotational acceleration around the centre of gravity. A mixture of both is also possible. For the model, a real-world boat, the “Messin” owned by the University of Rostock [87] is considered. The boat is shown in Figure 5.1 during a test in the Rostock harbour as part of the GALILEOnautic 2 project [52]. It is equipped for autonomous driving and is used as a testing vehicle throughout the project. The project also aims at equipping a real ship, the “DENEB”, for autonomous driving. However, when evaluating the algorithms in this thesis, no relevant measurement data existed for the DENEB. Thus the Messin was chosen for demonstrating the applicability of the investigated algorithms.

The POD drives of Messin are visible in the 3D-model in Figure 5.2. Messin has two propellers at the rear, which can be rotated around their vertical axis to steer the boat. An important assumption for the modelling is that both propellers operate symmetrically. This symmetry means that they have the same geometrical position relative to the centre of gravity, and they always cause an equal force in direction and intensity. Thus even if both POD drives can be operated independently, for the model in this thesis, they are considered as one single “virtual” actuator. All in all, this yields two controls for the Messin, the angle ϕ and the intensity E. The intensity is named E because it is also called the “Engine Order Telegraph” (EOT), a historically grown name. The EOT gives the intensity of the actuators as a percentage of the maximal possible value (“full ahead”).

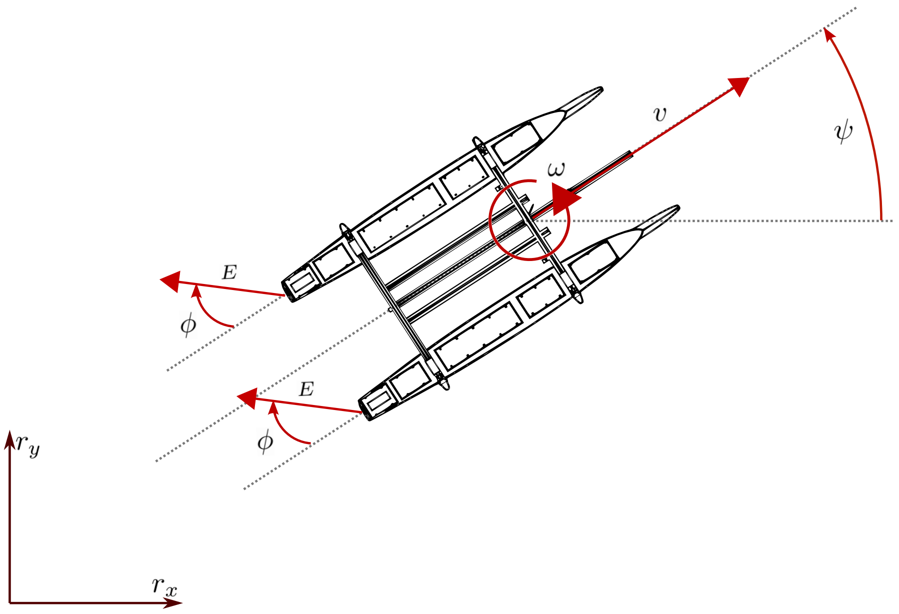


Figure 5.3: Graphical explanation of the Messin model and corresponding states. Underlying drawing from [116].

Figure 5.3 shows the details of the Messin-model. All values are represented by functions depending on the time $t \in \mathbb{R}$. The control $E(t) \in \mathbb{R}$ causes a force, for which $\phi(t) \in \mathbb{R}$ controls the direction. This force is split in its forward and rotational parts, which induces an acceleration in the respective direction. Those accelerations result in a forward velocity $v(t) \in \mathbb{R}$ and rotational velocity $\omega(t) \in \mathbb{R}$. Both are dampened by the water resistance, modelled by the cubic terms $p_2 v(t)^3$ and $p_4 \omega(t)^3$. The cubic approach makes sense since it preserves the sign. In contrast, a linear approach did not perform as well in the modelling process. Through a coordinate transformation from ship-centric coordinates to a fixed coordinate system, the movement within the coordinates $r_x(t) \in \mathbb{R}$ and $r_y(t) \in \mathbb{R}$ is computed. The model parameters $p_1, p_2, p_3, p_4 \in \mathbb{R}$ are used to represent several ships with the same model and need to be computed before the optimisation. All in all, this leads to the equations

$$\begin{pmatrix} \dot{r}_x(t) \\ \dot{r}_y(t) \\ \dot{v}(t) \\ \dot{\psi}(t) \\ \dot{\omega}(t) \end{pmatrix} = \begin{pmatrix} v(t) \cdot \cos(\psi(t)) \\ v(t) \cdot \sin(\psi(t)) \\ p_1 \cos(\phi(t)) \cdot \frac{E(t)}{100} - p_2\, v(t)^3 \\ \omega(t) \\ p_3 \sin(\phi(t)) \cdot \frac{E(t)}{100} - p_4\, \omega(t)^3 \end{pmatrix}. \tag{5.1}$$

States in this model are position $r_x(t), r_y(t)$ in meter [m], forward velocity $v(t)$ in meter per second $[\frac{\text{m}}{\text{s}}]$, heading of the ship $\psi(t)$ in radiants [rad] and rotational velocity $\omega(t)$ in radiant per second $[\frac{\text{rad}}{\text{s}}]$, and the controls: Engine Order Telegraph (EOT) $E(t)$ in percent [%] and propeller angle $\phi(t)$ in radiants [rad]. The EOT is a value representing the percentage from the maximum propeller speed where $E(t) = 100$ means "full ahead".

The goal is to steer the ship to a given final position and to stop at this position. However, it is assumed that for the nMPC setting, it is also necessary to compute a trajectory for a fixed time horizon t_f and the time interval $[0, t_f] \subset \mathbb{R}$, thus it might not be possible to reach the final position in the current iteration. To still be able to steer the ship towards the goal, the first objective is to minimise the distance of the last trajectory point $r(t_f) = (r_x(t_f), r_y(t_f))^\top$ to the final target position $r_f \in \mathbb{R}^2$.
For the second objective the norm values of the final velocities $|v(t_f)|$ and $|\omega(t_f)|$ are minimised. This minimisation addresses the idea that in the final position, a stable point must be reached. The minimisation also stabilises the behaviour of the optimisation modules in a way that if the final trajectory point is within a curve, that this point is not an open-end with too much movement, which might prove to be impossible to stabilise in the next nMPC iteration.

All in all this leads to the objective function

$$F = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} \|r(t_f) - r_f\|_2^2 \\ |v(t_f)|^2 + |\omega(t_f)|^2 \end{pmatrix}, \tag{5.2}$$

where the norms are squared to get differentiable functions. Note that here $|v(t_f)|^2$ and $|\omega(t_f)|^2$ are summed up to form one objective. This summation might seem counterintuitive since it is possible to add each term as an individual objective when dealing with multi-objective optimisation. However, in many settings, these goals do not contradict each other, which leads to unnecessary computations due to the higher dimension in the objective function, and since the computational effort grows exponentially, this can not be neglected.

Example 5.3 (POD-Drive Ship)
To get a finite dimensional optimisation problem, the example is formulated as a discretised optimal control problem using the trapezoidal scheme in the same way as for Example 3.3 in Chapter 3. With the definitions

$$\chi(t) \in \mathbb{R}^{N_\chi} := \begin{pmatrix} r_x(t) \\ r_y(t) \\ v(t) \\ \psi(t) \\ \omega(t) \end{pmatrix}, \tag{5.3}$$

and

$$u(t) \in \mathbb{R}^{N_u} := \begin{pmatrix} E(t) \\ \phi(t) \end{pmatrix}, \tag{5.4}$$

as well as the equations (5.1) and (5.2) the following optimisation problem is derived

$$
\begin{aligned}
\min_{\tilde{x}\in\mathbb{R}^{N_\chi(N_t+1)},\ \tilde{u}\in\mathbb{R}^{N_u(N_t+1)}} \quad & \begin{pmatrix} \left\|(\chi_1^{[N_t]},\chi_2^{[N_t]})^\top - r_f\right\|_2^2 \\ \left|\chi_3^{[N_t]}\right|^2 + \left|\chi_5^{[N_t]}\right|^2 \end{pmatrix}, \\
\text{subject to} \quad & \chi^{[i]} = \chi^{[i-1]} + \tfrac{h}{2}\left(f(\chi^{[i-1]},u^{[i-1]}) + f(\chi^{[i]},u^{[i]})\right),\ i=1,\ldots,N_t, \\
& \chi^{[0]} = (0,0,0,0,0)^\top, \\
& u^{[i]} \leq u_{\max}, \quad i=0,\ldots,N_t, \\
& u^{[i]} \geq u_{\min}, \quad i=0,\ldots,N_t, \\
& \left\|(\chi_1^{[N_t]},\chi_2^{[N_t]})^\top - r_f\right\|_2^2 \leq r_{\max},
\end{aligned}
\tag{5.5}
$$

where $N_t \in \mathbb{N}$ is the number of discretisation points, $u_{\min}, u_{\max} \in \mathbb{R}_+$ represent the minimal and maximal possible control value and $r_f \in \mathbb{R}^2$ is the target position. As in the Rocket Car example (Example 3.3) $\chi^{[i]} \in \mathbb{R}^{N_\chi}$ and $u^{[i]} \in \mathbb{R}^{N_u}$ represent the controls and states at the ith discrete point in time, while $\tilde{x} \in \mathbb{R}^{N_\chi(N_t+1)}$ and $\tilde{u} \in \mathbb{R}^{N_u(N_t+1)}$ are vectors formed by concatenating all discrete points. The function $f(\chi,u)$ represents the system dynamic formed by (5.1).
The last constraint is introduced to be able to force the ship into movement, since otherwise there exist several controls $u(t)$ including the constant control $u(t)=0$ which accelerate only a little at the beginning and then stop, which would all be weakly Pareto-optimal. Without the additional constraint the individual minimum for F_2 is also not strict since there are multiple controls which lead to $F_2 = 0$. This causes numerical problems when the individual minima are needed for a method. To circumvent this, the value of $r_{\max}$ needs to be set appropriately. In this example a good value is the one computed by the value of F_1 if F_2 is set to 0 as a constraint. Of course, this means additional effort for preparation.

In the next section, the algorithms presented in this thesis will be applied to Example 5.3 to see how they perform in a real-world problem.

5.2 Evaluation of the Performance of different Methods for Trajectory Optimisation Problems

This section applies the methods developed and improved in Chapters 4 and 3 to the trajectory optimisation problem illustrated by Example 5.3.
The structure of the section follows roughly the process a decision-maker would go through when investigating the multi-objective structure and properties of the problem. First, Section 5.2.1 shows how to compute the overall shape of the Pareto front through discrete approximations. For this computation, the Pascoletti-Serafini method described in Section 2.5.3 and the NsSQP algorithm, which Chapter 4 develops from the MOSQP algorithm are used. The section also shows results for the weighted sum method described in Section 2.5.1 to give a comparison with this simple method, thus providing an insight into the tradeoff between the simple implementation of the weighted sum and the better theoretical properties of the MOSQP and Pascoletti-Serafini approaches.

After the general approximation of the Pareto front, this section aims at reducing the samples needed for an accurate approximation. For this, Section 5.2.1 shows how the algorithms mentioned above perform if the number of samples is reduced. Thereafter, Section 5.2.2 applies the

p_1	0.1874
p_2	0.0350
p_3	0.2137
p_4	7.8112

Table 5.1: Model parameters used within the optimisation of Example 5.3.

interpolation techniques based on parametric sensitivity analysis developed in Chapter 3 to the problem. The section shows how the additional information is utilised to get more information out of one single sample and explains how this information aids in developing a controller for the example problem 5.3.

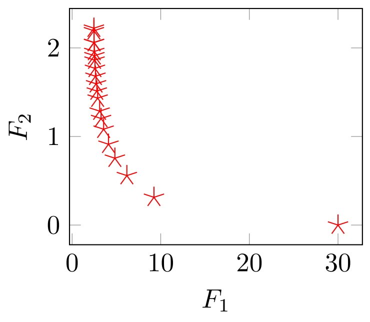
Note 5.4
This section aims at investigating the general properties of the different methods. Thus the comparison will be mainly focused on the results of the approximation of the Pareto front. Performance in terms of computation time will only be investigated briefly. A detailed comparison based on computation times would require a detailed tuning of the used hyperparameters and the written code to ensure that the abilities of the methods are compared and not only some implementation details.

Note 5.5
All computations for this section are done on a “Lenovo Thinkpad T14” running “Ubuntu 20.04.2 LTS (64 bit)” as the Operating System with a “AMD® Ryzen 7 pro 4750u with Radeon graphics × 16” processor and 32 GB memory. The used WORHP parameter settings are provided in the respective xml files in [4]. For all computations the number of discrete time points is set to $N_t = 100$. The model parameter are computed based on data from the project GALILEOnautic 2 using the tool TOPAS-Model-Fitting from [124]. The values are set as in Table 5.1.

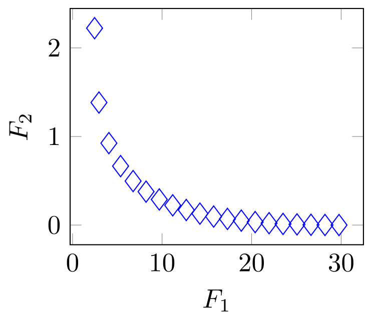
5.2.1 Discrete Approximation for the Pareto Front of an Autonomous Ship Controller

When approaching the solution of a multi-objective optimisation problem, the general shape of the Pareto front provides very helpful information. This shape will give, for example, the information whether or not the Pareto front is convex, which helps a lot when choosing the solution algorithm for future applications. Thus, this section first evaluates how well the different algorithms compute the overall shape. For this, 20 samples are computed with each algorithm.

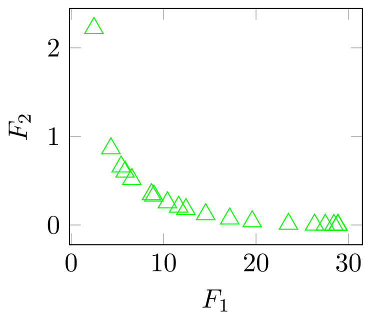
Figures 5.4a to 5.4d show the resulting Pareto fronts computed with the weighted sum scalarisation, the Pascoletti-Serafini scalarisation, the NsSQP algorithm and the MOSQP algorithm, respectively. The x-axis represents the distance to the target position r_f and the y-axis the velocities in the last trajectory point as described in Equation (5.2). All results have in common that they can represent the overall shape of the Pareto front. Each result spans the same interval of solutions. However, this is expected since all of them use the individual minima to determine this interval. It is also possible to infer the convexity of the Pareto front from each result and make general statements about tradeoffs. For example, the nMPC developer can conclude from all figures that the movement in the final position can be decreased by roughly 2/3 by simultaneously increasing the distance to the target position by only roughly 1/6. The main difference between the individual results is in the distribution of the samples.



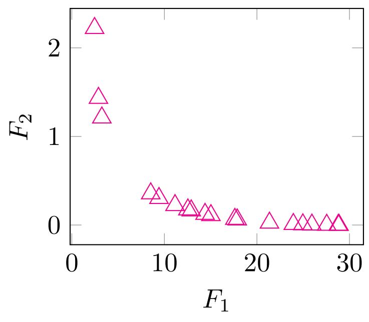
(a) Weighted sum scalarisation.



(b) Pascoletti-Serafini scalarisation.



(c) NsSQP algorithm.



(d) MOSQP algorithm.

Figure 5.4: Discrete approximation of Example 5.3 computed with 20 samples.

As expected, the result from the Pascoletti-Serafini approach shown in Figure 5.4b is the most evenly distributed. The distances between neighbouring samples are almost equidistant over the whole set. Note that the scaling of the axis explains the gap in the top left section and that the distance between two points is only equidistant when taking both dimensions into account. Thus distances between two samples will possibly vary when evaluated for each dimension individually. In contrast, the samples shown in Figure 5.4a which are computed by using the weighted sum, are more concentrated on the left side of the Pareto front. In this result, the density of samples decreases from left to right. Between $F_1 \approx 10$ and $F_1 \approx 30$ there exists also one big gap. Thus, the samples only cover $1/3$ of the Pareto front, with the individual minima as the exception. Although it is still possible to depict the general shape of the Pareto front, this gap nevertheless poses a general problem when using the weighted sum. The gap is caused by changing the weights only by the amount of 0.05. This behaviour is a problem because it leads to unintuitive behaviour when only varying the weights without sampling the whole Pareto front. It also shows that the solution of the scalarised problem is in the right section and is very sensitive to the weights. This needs to be taken into account if the nMPC module should change the weights in some situations during the real-time process.

The results from the NsSQP and MOSQP algorithms are shown in Figures 5.4c and 5.4d respectively. As the other results, both provide a good overview of the overall shape of the Pareto front and are able to display the convexity and the general tradeoffs.

For both algorithms, the resulting samples are mainly distributed around the lower right section of the Pareto front, which is, however, due to scaling a good distribution over the whole front, similar to the Pascoletti-Serafini results. This is interesting since the NsSQP and MOSQP algorithms use problems internally which are more related to the weighted sum. Thus, the fact that these algorithms can compute samples in the region where the weighted sum itself is highly sensible and thus prone to gaps is a significant result. Their most significant difference to the result from the Pascoletti-Serafini algorithm is the variance in the distribution. So while they compute an excellent overall spread, the size of each gap varies visibly. In the results of both algorithms appear at least two samples that are almost directly next to each other. In contrast to the weighted sum, the pattern in which the gap size varies is more or less random, whereas the weighted sum displays increasing gap sizes from left to right.
Compared only with each other, the results for the NsSQP and the MOSQP display minor differences. For example, while both have one bigger gap in the Pareto front, they are at different positions. Likewise, the distribution of the individual samples is a bit different such that there is no clear region where both have a higher density. However, when the scaling of the axis is taken into account, the gap in the result of the MOSQP algorithm is bigger in total numbers than the one of the NsSQP algorithm. The NsSQP algorithm produces almost the same gap as the Pascoletti-Serafini result. Thus overall, the spread of the NsSQP could be considered slightly better than the one of the MoSQP. However, if this is a structural advantage of the modification or just a random appearance can not be concluded from this one example (compare Section 4.7). This becomes even more obvious when considering that the motivating situation for the modification of the cleanup streategy in the MOSQP algorithm is the situation where the Pareto front is not convex does not apply to this example, the performance of both algorithms can be considered almost the same in this situation.

There is also one difference between the weighted sum and the Pascoletti-Serafini to the NsSQP and MOSQP algorithms which can not be seen in the results. This difference is that the former depends on a hyperparameter that needs to be changed to find a distribution of samples of the Pareto front. The latter, in contrast, work without a hyperparameter.
However, for the weighted sum, it is no problem to determine a suitable distribution of weights. Of course, this only holds when the situation is such that the weighted sum can find all points of the Pareto front. For the Pascoletti-Serafini approach, some computations have to be performed to find a good distribution of points (compare Section 2.5.3), especially in dimensions $N_F > 2$ this is a non-trivial task. But in the setting investigated in this section, it is not a problem since it only involves the computation of individual minima which already form a part of the Pareto front.
In contrast to the negative interpretation of the necessary hyperparameter, it can also be seen as a positive attribute. For example, the possibility of limiting the parameter range gives the decision-maker another tool. Thus, it would be possible to further refine a specific region after a first rough computation. With both the weighted sum and the Pascoletti-Serafini approach, it is easy to choose hyperparameters such that only samples left or right of an already computed sample are found. For the NsSQP and the MOSQP, this would introduce the necessity of adding another constraint to the problem. However, this additional constraint increases the problem dimension, can influence numeric behaviour, and has, in general, a more substantial impact on the problem.

The computation of the 20 samples for the results in Figures 5.4a to 5.4d took a very different amount of time for each method. Table 5.2 shows that the Pascoletti-Serafini approach was the fastest, followed by the weighted sum, which already took twice the time for computing

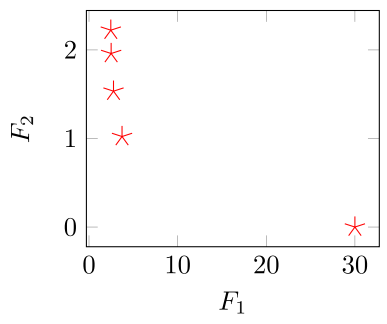
Method	Time [s]
Pascoletti-Serafini	67.613
Weighted sum	144.277
MOSQP	316.500
NsSQP	375.984

Table 5.2: Computation time for Pascoletti-Serafini, weighted sum, MOSQP and NsSQP on the POD-Drive ship Example 5.3 based on computing 20 samples.

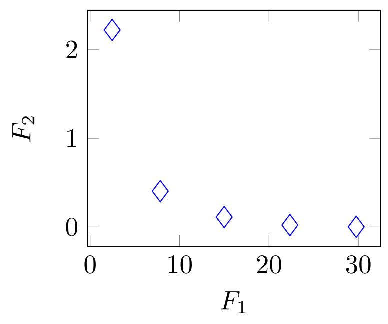
20 samples. The MOSQP and NsSQP methods took again twice as long as the weighted sum to compute the samples. Thus the rough first impression is that the Pascoletti-Serafini method performs the best and the NsSQP and MOSQP pay for their independence of a-priori parameter with computation time. However, if, for example, the Pascoletti-Serafini always performs better than the weighted sum, or if it just depends on the example and WORHP-parameter setting, would need to be investigated further. In addition, the MOSQP and NsSQP algorithms spend most of the computation time in the spread phase (see description of the algorithm in 4.13). However, the number of iterations in this phase can be varied arbitrarily, which strongly influences the computation time. One observation done for the investigated example is that almost no points are stopped early within the spread phase. Also, the MOSQP and the NsSQP waste computational power on solutions which are discarded afterwards in the cleanup stage.
Note that for all four methods, some effort for parallelisation was made in the implementation. Thus, the huge differences probably are not caused by this computation of discarded solutions. However, they differ in the details of their parallelisation. For example, the weighted sum can compute all samples completely parallelly, whereas the Pascoletti-Serafini method needs to solve the optimisation problems for the individual minima first and has a synchronisation point after this phase. In contrast, the MOSQP and NsSQP algorithms have synchronisation points after each phase and each major iteration in the spread phase, influencing performance.
Besides the comparison between each method, the computation times in Table 5.2 show that it is necessary to make some more effort to reduce computation times to use a Multi objective algorithm within the real-time environment of nMPC applications. One possibility is to reduce the number of samples. Next, this section investigates the impact of such a reduction of samples onto the approximation of the Pareto front.

So far, the general ability of each algorithm to find an approximation of the Pareto front with a relatively high number of points was investigated. However, often it is necessary to reduce the number of samples. This reduction may be necessary because each sample takes much time to compute, because the dimension N_F increases, or because many different Pareto fronts need to be computed for many situations. It is also possible that the resources available to compute the Pareto front decrease. This decrease might be because there is only very little time or a very limited computational power. Both limitations are given when switching to real-time embedded nMPC computations.
The reasons presented above motivate a general interest in decreasing the number of samples needed to get a good approximation of the Pareto front. Thus, the next step in this section is to investigate how the results shown in Figures 5.4a to 5.4d change if the number of samples is reduced.

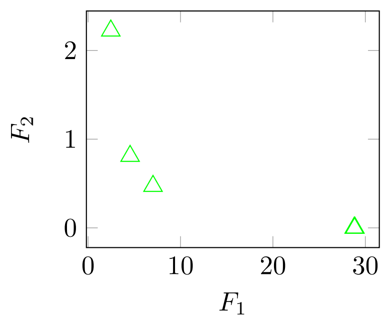
For the results presented in Figures 5.5a to 5.5d only 5 samples were used to compute the



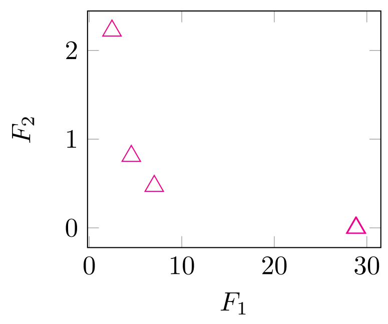
(a) Weighted sum scalarisation.



(b) Pascoletti-Serafini scalarisation.



(c) NsSQP algorithm.



(d) MOSQP algorithm.

Figure 5.5: Discrete approximation of Example 5.3 computed with 5 samples.

Pareto front. When comparing the results of the weighted sum and Pascoletti-Serafini in Figures 5.5a and 5.5b with the results above in Figures 5.4a and 5.4b where more samples were taken, the behaviour stays almost the same. The samples computed with the weighted sum are concentrated in the left section of the Pareto front. In contrast, the samples from the Pascoletti-Serafini method are distributed equidistant over the whole Pareto front. However, if one takes the scale of the objectives into account, the approximation in Figure 5.5b shows again a gap on the left side. In terms of numbers the weighted sum computes samples in roughly $^1/_{30}$ of the interval of F_1 and $^1/_2$ of F_2 (excluding the extrema), the Pascoletti-Serafini method, in contrast, produces samples in $^{25}/_{30}$ of the interval of F_1 and $^1/_4$ of F_2.
However, one big difference between both methods is that the behaviour of the Pascoletti-Serafini scalarisation is predictable, and therefore this gap due to different scales should be expected. All in all, one can give a resolution in absolute values, and the method will produce the respective approximation. The weighted sum, on the other hand, is less predictable. Although the behaviour is not surprising since it is well known for the weighted sum, it is unintuitive. It is hard to predict how the resolution will be, making it harder to improve the performance by reducing samples. It is harder to refine an already computed Pareto front since it is not directly possible to compute another sample in the "middle" of two existing ones, which is relatively easy with the Pascoletti-Serafini approach.
For NsSQP and MOSQP the results are shown in Figures 5.5c and 5.5d. It can be seen that the results are the same for both methods. This is not surprising for two reasons: First, as

already stated for the results in Figures 5.4c and 5.4d the motivating situation for the modification towards the NsSQP does not apply here. Second, the difference between the methods will only influence the algorithm if at some iteration the set of computed points has less than $\bar{n}$ nondominated points, with $\bar{n}$ the maximal number of points allowed in the intermediate sets. If $\bar{n}$ is low, as is the case here, it is likely that there are always $\bar{n}$ nondominated points in the set. At first glance, the computed approximations of the NsSQP and the MOSQP are almost symmetrical. However, this once again holds only when the scale of the objectives is not taken into account. On a closer look, the MOSQP and NsSQP algorithms compute approximations similar to the weighted sum. This is because all samples except the individual minima of F_2 have a value $F_1 < 10$ and thus cover only a tiny section of the Pareto front.
Compared to the results of the Pascoletti-Serafini and weighted sum approach, an essential difference is that the NsSQP and MOSQP algorithms only compute four different samples instead of five. This can happen since duplicates are removed during the iteration. However, the overall shape of the Pareto front can still be deduced from the four samples computed.

All in all, each of the algorithms can compute an approximation that illustrates the main shape of the Pareto front. Thus, even though the differences can not be neglected, the decision-maker would get a rough overview of the Pareto front from all four approaches. One significant difference is how the decision-maker could proceed afterwards. As mentioned above, it is impossible to refine a specific region of the Pareto front with the NsSQP and MOSQP algorithms without introducing new constraints. Even if this is simple, it is hard to control where exactly the refined samples will be. For the weighted sum and the Pascoletti-Serafini approach, this is possible by restricting the respective hyperparameter. The most precision for this is possible with the Pascoletti-Serafini approach, especially when applying a refinement method like proposed by Eichfelder [33]. For the weighted sum, it is possible to control the range of solutions via the hyperparameter, but similar to NsSQP and MOSQP, it is not immediately clear where precisely the newly computed samples will lie.

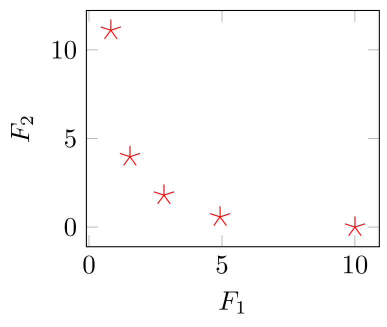
Since most of the observed behaviour appears to be coupled to the scale of the individual objectives, it is interesting to see how the methods behave if scaled to roughly the same interval. For this, the objective function

$$\tilde{F} := \begin{pmatrix} \frac{1}{3}F_1 \\ 5F_2 \end{pmatrix}, \tag{5.6}$$

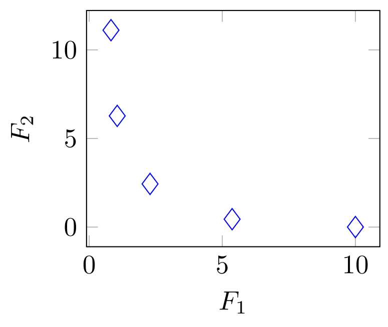
with F from the Example 5.3 is used for the optimisation.

Note 5.6
The reformulation done in Equation (5.6) does usually not scale the objectives to the same interval. For this, a shift would be necessary, too. In addition, the scaling factors are not the exact maxima. However, the focus here is also on simplicity, and for the given situation, the proposed scaling is sufficient.

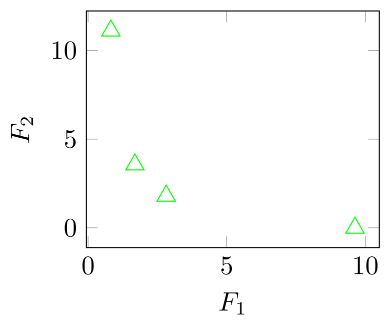
By scaling the objective functions as defined in (5.6) the results in Figures 5.6a to 5.6d are computed. For the results of the weighted sum and Pascoletti-Serafini scalarisation, the scaling effect has a high impact on the distribution. The results are shown in Figures 5.6a and 5.6b. Especially for the Pascoletti-Serafini approach the scaling causes an almost perfectly equidistant distribution in both objectives. This is the expected behaviour and outlines the advantage of the Pascoletti-Serafini approach that the distribution of samples is very predictable.



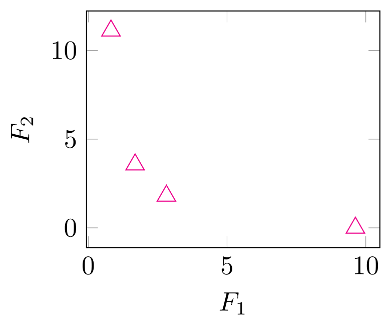
(a) Weighted sum scalarisation.



(b) Pascoletti-Serafini scalarisation.



(c) NsSQP algorithm.

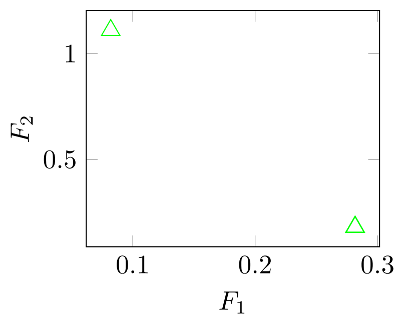


(d) MOSQP algorithm.

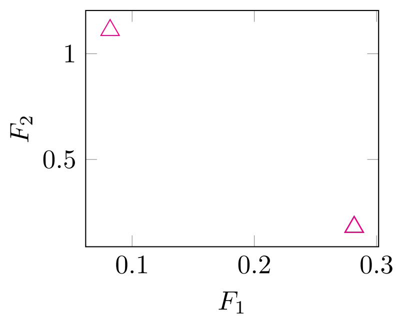
Figure 5.6: Discrete approximation of Example 5.3 with scaled objective functions (5.6) computed with 5 samples.

The impact of scaling on the weighted sum is also high but does not lead to an equidistant distribution. As shown in Figure 5.6a the samples are now distributed symmetrically but are also concentrated around the middle section of the Pareto front. Although the distribution gives a comparable overall impression of the Pareto front, the results show how hard it is to predict the weights' influence in the weighted sum even if scaled almost perfectly.

One interesting fact is that the results for MOSQP and NsSQP show no visible differences when plotted. Of course, the distribution is now symmetrical when looking at the absolute values. However, the relative positions within the individual objective ranges do not change. When looking at the cleanup strategy in Section 4.2, this can be explained by the fact that Pareto-dominance itself is scaling invariant, and the crowding distance (see Section 4.2) is computed separately for each objective. Thus it is reasonable that a relative scaling of the objectives to each other has no impact. In addition, the problems solved during the spread phase (see Equation (4.14)) also only involve each objective individually. In fact, the first and only point where the objectives are evaluated against each other is in the refine phase (see Equation (4.32)). However, the optimisations within the refine phase may be dominated by the additionally introduced constraints. Thus it is possible that scaling of objectives does not change the relative positions of the samples within the Pareto front. It should be noted, however, that this is not a statement about scaling invariance. The refinement phase still includes a sum of all objectives. Furthermore,



(a) NsSQP algorithm.

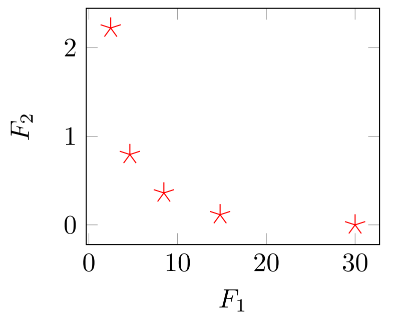


(b) MOSQP algorithm.

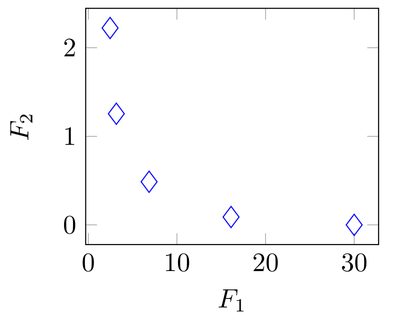
Figure 5.7: Discrete approximation of Example 5.3 with objective functions scaled to 1 computed with 5 samples.

the problems in Equation (4.17) in the spread phase are usually sensitive to a scaling since the scale of the objective influences the solution for numerical reasons (see for example [120, p. 125]).

If thinking about scaling for the MOSQP and NsSQP algorithm, it is worth looking at the results if the objectives are scaled to 1. The results for this case are shown in Figures 5.7a and 5.7b. It is shown that the algorithm is not able to compute a well-spread distribution. Even the individual minima for F_2 is "destroyed" during the iterations. This destruction happens during the refinement stage, where all samples are optimised using the same objective function. The hope here is that the constraints will prevent the samples from converging to one single point. However, if the points are not close enough to the Pareto front after the spread phase, it can still happen that they converge to one single point. That this happens to the individual minima too is, in fact, interesting. It might be due to numerical imprecision in the initialisation phase or due to a very flat Pareto front.



(a) Weighted sum scalarisation.



(b) Pascoletti-Serafini scalarisation.

Figure 5.8: Discrete approximation of Example 5.3 with scaled objective functions (5.6) rescaled after optimisation computed with 5 samples.

All in all, scaling has a significant impact on the approximations computed by the different scalarisation approaches. However, the scaled results provide only information about relative values. If the decision-maker is interested in absolute values, the results have to be scaled back. That the good distribution of the weighted sum and Pascoletti-Serafini scalarisations are not affected in this case is shown in Figures 5.8a and 5.8b. They show that the distributions almost stay the same if rescaled to the original values. Thus it is possible to use the scaled results for statements about both relative and absolute values.

In conclusion, all applied solving techniques can compute the overall shape of the Pareto front and provide critical first information to the decision-maker. The shape of the Pareto front in the investigated example is convex. Differences between the solvers are mainly in the uniformity within the distribution of samples, the dependence on hyperparameters and the computation time. Scaling affects all of the solving methods positively but needs to be applied with some caution since the MOSQP and NsSQP algorithms show problematic behaviour if the range of the objectives becomes too narrow.

5.2.2 Approximating the Pareto Front of an Autonomous Ship Controller Continuously by the Application of Parametric Sensitivity Information

The last section investigated how each solver performs in approximating the Pareto front of the nMPC-Example 5.3 when the decision-maker looks for an approximation that is only based on the samples information. This section applies the methods described in Chapter 3 to the example to utilise parametric sensitivity information within the development process of an nMPC module for autonomous ships. The goal is to investigate which additional information which this technique can provide to the decision-maker and thereby reduce the number of samples needed for an accurate approximation of the Pareto front. Problems and pitfalls are also discussed throughout the section. First, the methods to provide locally continuous approximations around each sample from Section 3.1 are investigated. Subsequently, it is shown how global approximations as described in Section 3.2 help in the setting of the nMPC-Example.

Local Approximation

Figure 5.9 shows the local approximation around 3 samples computed with the Pascoletti-Serafini approach. For comparison, the Pareto front is also sampled with a high resolution which is shown as green stars. The green samples are considered to be the true Pareto front in this example in contrast to the approximations. The figure shows that not all green samples are within the Pareto front. This shows the significant problem of numerical artefacts when sampling the Pareto front. Especially the low-pass filter feature of WORHP can lead to samples that are stopped too early, but also other problems like too high tolerances may lead to early stopped optimisations. Thus it is important to filter dominated points from the computed samples. In the results presented here, they were left in to illustrate this problem. Note that one of these artefacts is directly next to one of the samples used for local approximation. Thus it is only due to luck that such an artefact does not destroy the approximation.
Around the samples, which are displayed in blue, are the continuous linear and quadratic approximations shown in orange and magenta, respectively. The orange lines are twice as thick as the magenta-coloured ones to make the linear approximations better visible.

The results show that with the local approximation done via the parametric sensitivities as

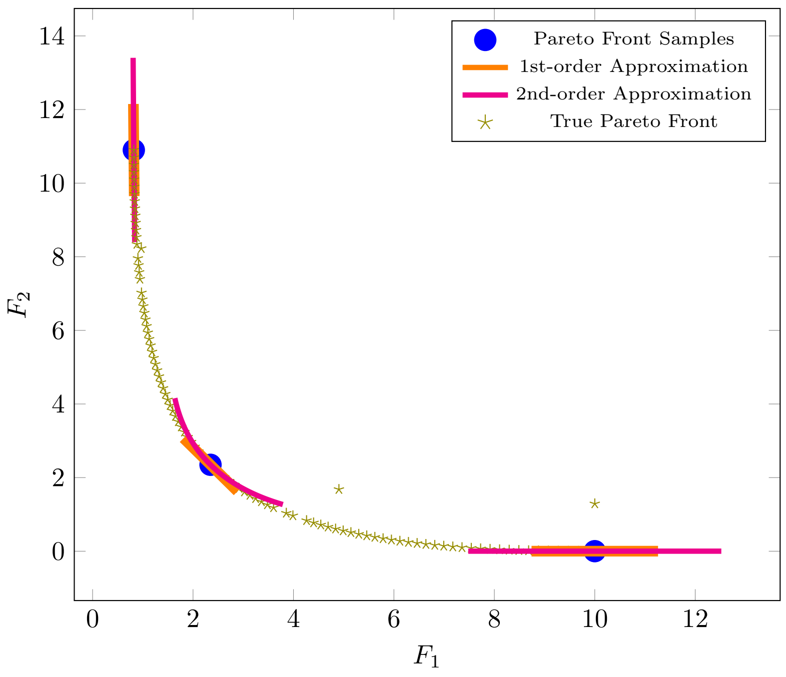


Figure 5.9: Continuous local approximation at each sample based on objective space parametric sensitivity information for the Example 5.3 with scaled objective functions (5.6).

described in Chapter 3 it is possible to get an excellent overall impression of the Pareto front with only three samples. Even more important is the information available by each sample individually. When considering, for example, the middle sample, the decision-maker directly gets the information that the derivative at that sample is almost -1. Thus the tradeoffs between each objective are even. This information means that if one objective is decreased by 1%, the other one would need to be increased by roughly 1%.

In contrast, the outer samples show tradeoffs where one objective could be decreased with almost no impact on the other. Samples like these are usually considered to be less favourable, which is also covered in the concept of proper Pareto-optimal points (see [91]). By using the parametric sensitivity analysis, the decision-maker gets these insights without computing more samples.

Of course, this is only local information and only valid in a specific area around the sample. To extend the knowledge about tradeoffs along the Pareto front, the curvature becomes of interest because the curvature and thus the quadratic approximation provides information about how the tradeoffs will change along the Pareto front. Especially if the curvature is 0, as is the case for both outer samples, it is clear that the tradeoffs will stay the same locally, which hints at a locally flat Pareto front.

If the decision-maker gets all three samples with the sensitivity information available, they are evaluated concerning the information mentioned above. In the situation displayed in Figure 5.9, this will usually lead to the insight that the sample in the middle is the most favourable. Of course, this selection is also possible in this scenario when looking only at the three samples themselves. However, using the sensitivity analysis, it is also possible to make this decision when looking only at one sample.

For example, if the scalarised optimisation produces the sample on the top left, the parametric sensitivities show that the tradeoffs are very uneven and will not change locally. Thus the

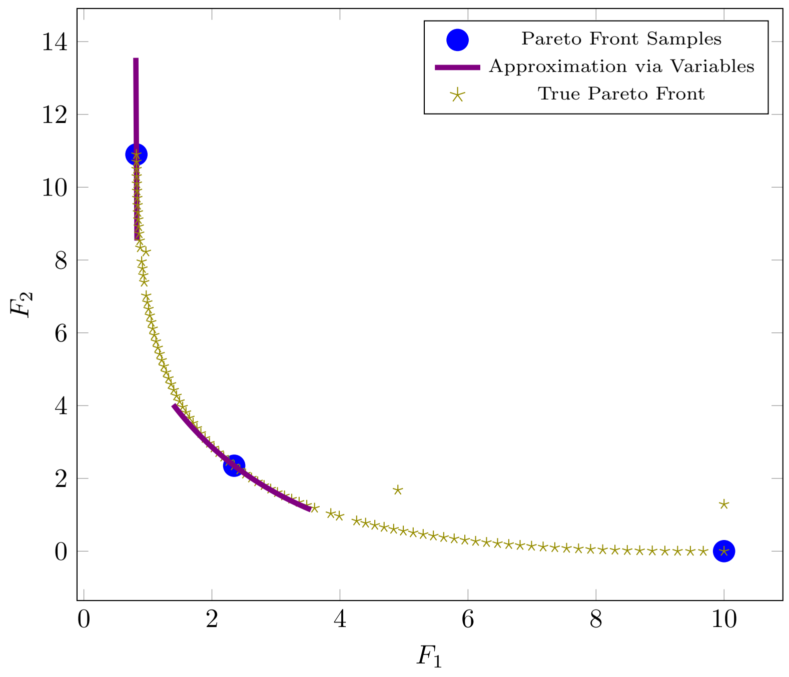


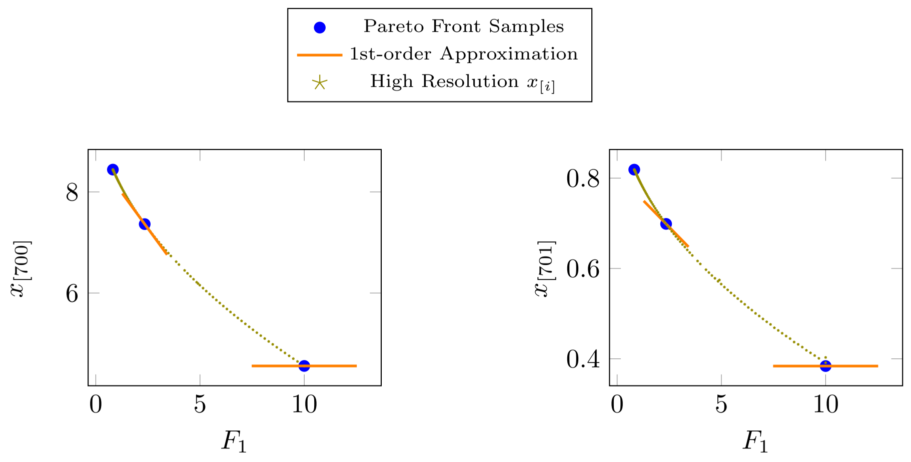
Figure 5.10: Continuous local approximation at each sample based on variables parametric sensitivity information for the Example 5.3 with scaled objective functions (5.6).

decision-maker could adjust the hyperparameter using the parametric sensitivity analysis again as explained in [33]. If the second optimisation computes the sample in the middle, the information available shows that the sample is favourable in terms of even tradeoffs.
The described procedure can be used in Pareto front navigation [1] or even be utilised by an automated module that steers the multi-objective optimisation process within an nMPC module.

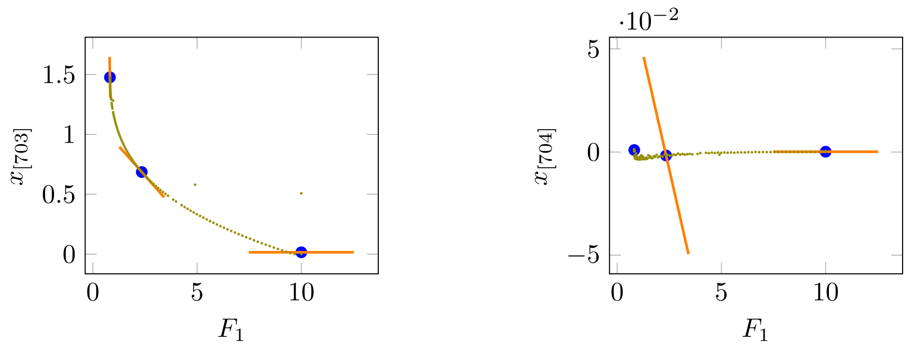
Next, it is investigated if the sensitivity analysis for the variables can lead to comparable results. This is especially of interest since Section 3.3 shows that bang-bang controls might lead to problems when approximating the variables. In addition, the sensitivities of the variables are of particular interest for nMPC settings since they make it possible to adapt the optimal solution to small changes in real-time. This adaptation could be utilised when varying a solution around a sample of the Pareto front without recomputing it entirely.
Applying the technique described in Section 3.1.2, where the objective function is evaluated for the approximated variables, leads to the results shown in Figure 5.10. As before for the approximation in objective space in Figure 5.9, the samples computed by the Pascoletti-Serafini approach are shown as blue dots and the higher resolution Pareto front as green stars.
Compared to the results from before, the approximations around both left samples show the same behaviour. The decision-maker gets insight into the unfavourable tradeoff at the left border. The approximations also yield the tradeoff and curvature at the middle sample in the same way as before. Thus the same information about tradeoffs can be retrieved by those two samples alone. Note that the nonlinear behaviour at the middle sample is approximated accurately by using only linear approximations, which is a remarkable advantage of the approximation via variables.

One big difference is the lack of any approximation around the right sample. This is due to a problem which was not visible in the results in Figure 5.9. At the border of the Pareto front, slight numerical differences can lead to problems with the sensitivity information. For example, in the situation displayed here, the constraint which restricts F_1 can become active. In this case, the Pareto-optimal solution is not exactly at the intersection of the feasible set and $tr + a$ from Equation (PS(a, r)). Thus not all Pascoletti-Serafini specific constraints are active, which renders some of the parametric sensitivities useless. It can also lead to sensitivities equal to zero, which is the case here.
For the interpolation done directly in objective space, a sensitivity equal to zero leads to a constant approximation of one objective. Figure 5.9 displays this for the right sample. When using the sensitivities for the variables, a value of 0 causes constant approximations of the variables. This behaviour is visible Figures 5.11a to 5.11d. The figures display the local approximation of the variables on which the objectives depend explicitly. A look at the right sample reveals the constant approximation of each variable. At the same time, it is clear that this constant approximation is not the derivative of the variables, which becomes evident when comparing the approximations in orange with the true Pareto front in green.
When evaluating the objective function with the approximated variables in Figures 5.11a to 5.11d, no change will be visible. This situation is fixed by narrowing the interval in which the parameter a is chosen. However, this involves a recomputation of at least one sample on the right. It is also possible to choose a such that it will lead to the same solution but with both constraints active (see [33]). To get the sensitivities, it is still necessary to recompute the sample with the changed value of a. However, if reusing the solution as an initial guess, the recomputation is usually not expensive.
For the display in Figure 5.10 the fixes where not applied on purpose. The result shows possible pitfalls when using parametric sensitivities in real-world applications. Interestingly, this problem is directly visible when approximating the variables and not when approximating the Pareto front directly in objective space. This visibility can be considered an advantage of the approximation via variables. However, the resulting approximation in objective space in Figure 5.9 at the right sample is the expected one. Thus the objective space approach might be considered more robust. However, it is most likely a coincidence since the Pareto front happens to be almost constant at that sample. In addition, using the sensitivities of an inactive constraint at all is not guaranteed to work since the theory in Chapter 2.2 only applies to active constraints. To prevent using unreliable information, it is necessary to check if the respective constraints are active. If an inactive constraint is found, a recomputation with the method described in [33] is recommended.

Looking at the local approximations of the variables themselves shown in Figures 5.11a to 5.11d reveals also some more interesting facts. One is that the almost equidistant distribution of the samples in objectives-space does not lead to an equidistant distribution in variable-space. Of course, this was never expected, but if the focus is on the variables, this fact should be considered when choosing the solving algorithm, especially when using the variable approximations for a global approximation of the Pareto front as is described in Section 3.2 an equidistant distribution of the samples in variable space would be most beneficial. This equidistant distribution could be achieved by using the parametric sensitivities of the variables and by applying the adaptive approximation technique from Eichfelder [33] in variable space. However, Figures 5.11a to 5.11d show that the distribution is different for each variable, thus it would be necessary to find a compromise between all variables or to prioritise one variable.
Another interesting fact is that the sensitivity of the rate of turn at the middle sample seems to be incorrect at first glance. Figure 5.11d shows that the local approximation in orange is almost



(a) x-Position. (b) y-Position.



(c) Velocity. (d) Rate of turn.

Figure 5.11: Continuous local approximation of states in the final discrete point $\chi^{[N_t]}$ at each sample based on variables parametric sensitivity information for Example 5.3 with scaled objective functions (5.6).

perpendicular to the graph of high-resolution samples of the variable in green. The first guess when trying to explain behaviour like this is most often numerical issues. This assumption is backed up here because the high-resolution samples in the green sample left of the middle are not absolutely smooth. This observation leads to the assumption that the solving algorithm might not converge correctly here. The fact that the values for the rate of turn are lower than those for position and velocity by a factor of almost 10^{-2} might be a numerical challenge in this case. If the tolerance parameters for WORHP are not configured correctly, this difference in magnitudes might lead to problems. However, a closer look at the high-resolution samples in Figure 5.11d gives a hint that there might be a high-frequency behaviour of the Pareto front not captured correctly and that the seemingly incorrect local approximation might be caused by something comparable to the problems with the bang-bang control in Section 3.3. A very high resolution of the rate of turn on the Pareto front is needed to investigate this.

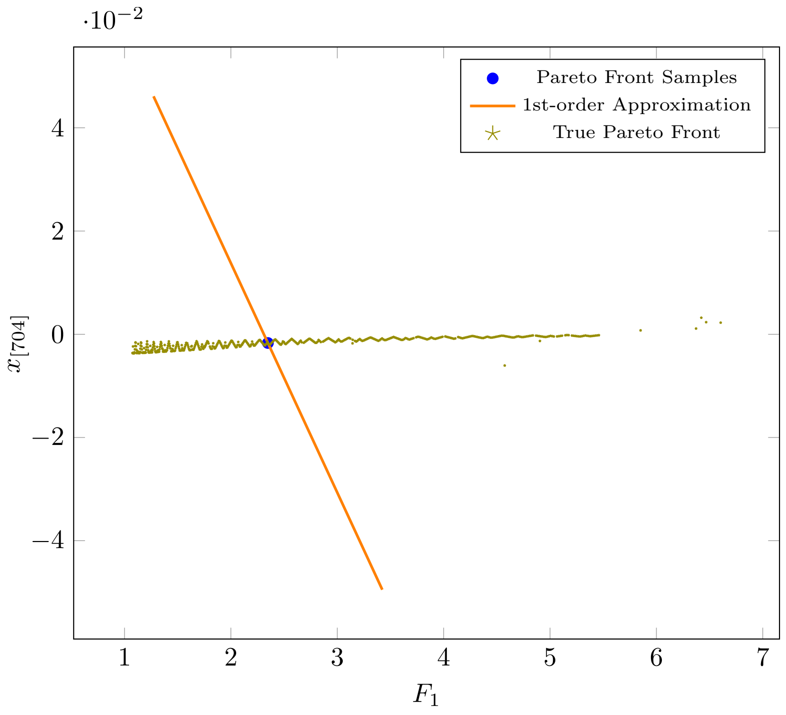


Figure 5.12: High resolution approximation of the rate of turn in the last discrete point $\chi^{[N_t]}$ along the Pareto front of Example 5.3 with local approximation around one sample.

Figure 5.12 shows the resulting rate of turn in the final discrete point with higher resolution around the middle sample of Figure 5.11d. The samples in green reveal that, indeed, the rate of turn has a high-frequency behaviour along the Pareto front. It also shows that the parametric sensitivities and thus the local approximation in orange is correct. This behaviour, once again, emphasises the locality of the parametric sensitivities. In contrast to the bang-bang control in Section 3.3, the local approximation is not even close to the overall trend of the variable and thus only valid in a very close neighbourhood of the sample. All in all, at this point, it is useless for the intended usage. However, it reveals a feature of the rate of turn along the Pareto front, which would probably have been overlooked if only a few samples had been taken without looking at the sensitivities. This behaviour might provide helpful insight into the general behaviour of the example.
Note also that the incorrect approximation of the rate of turn has no substantial influence on the approximation of the objective function. This is due to the scaling and that the small values of the rate of turn vanish in relation to the velocity, especially when squared.

Another essential difference between approximating in the objective space and approximating via variables is not visible in the Figures. This difference is that approximating via variables does not yield a single number for the derivative and curvature of the Pareto front as described in Section 3.1.2. Thus it is necessary to evaluate the objective function several times. If visualised, the numbers can then be guessed by inspecting the graph. If it is necessary to get exact numbers,

it is also possible to compute them with finite differences. In general, this is a disadvantage of the approximation via variables. However, in the situation investigated here, it does not lead to significant differences. Evaluating the objective function in Example 5.3 several times to get a visualisation is cheap, and the precision of finite differences is enough if a number is needed. Thus both approaches are equally applicable in this situation.
In general, optimisation problems resulting from discretised optimal control problems may have an expensive objective function, for expample, if the objective function consists of an integral term for a long time horizon. In this situation, the approximation via variables might get significantly more expensive. However, since the objective function is more complex, the method might also be advantageous because it preserves more nonlinearity than the direct approximation of the Pareto front. This preservation happens because the original nonlinear objective function is evaluated based on approximated variables.
It is also necessary to consider that computing the sensitivities for many variables is significantly more expensive than computing them for only two objectives. This is especially true if compared to the 1st-order information, which is just the Lagrangian multiplier λ and does not require any computation at all. On the contrary, the sensitivity information of the variables can also be reused for other purposes like fast Pareto front navigation or a real-time update of an offline computed trajectory. In this situation, the sensitivity information would need to be computed in any case.
In conclusion, it depends on the exact situation and usecase which of the methods is better suited. They can, of course, also be used as complementary information.

All in all, the parametric sensitivity analysis can produce valuable additional information about the Pareto front for each sample. It is particularly valuable when evaluating the samples separately, either because a large, complicated Pareto front is only sampled with a small number of samples, or even if only one sample is computed and the decision needs to be made if this sample is good or an adjustment of hyperparameter is favourable.

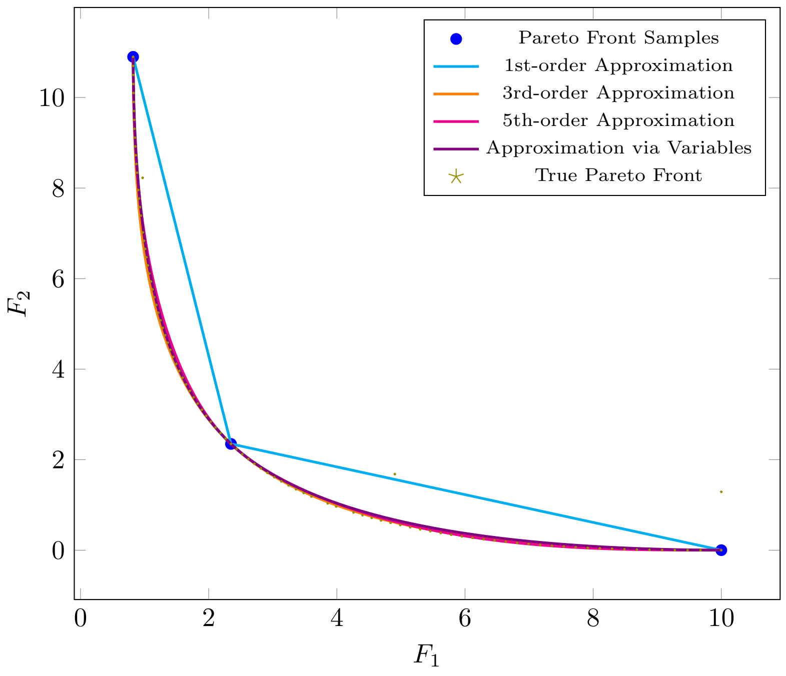


Figure 5.13: Continuous global approximation based on parametric sensitivity information of 3 samples for the Example 5.3 with scaled objective functions (5.6).

Global Approximation

Section 3.2 describes how the parametric sensitivities can be used to provide a smooth approximation of the whole Pareto front. This smooth approximation is made by forming polynomials of 1st-, 2nd-, and 5th-order with the information from the samples and the parametric sensitivities at the samples. For the nMPC setting, these approximations are of interest for possible applications, since they enable further reduction of the needed samples. However, Section 5.2.2 shows that it is already possible to get a very good overview over the Pareto front using the local approximations. Thus in real-world nMPC, the global approximation might not be necessary for this purpose. However, when the decision-maker is not a human but a computer algorithm, it is definitively advantageous to have a smooth overall approximation for the computer to be able to compute continuous values. This section will also show that with the global approximation, the samples needed to approximate the Pareto front can be reduced further.

To create the global approximation, the same three samples as for the local approximations are used. Figure 5.13 displays the resulting approximations. In light blue is the linear approximation, which uses only the samples themselves and represents the linear approximation that is performed when no sensitivity information is available. The 3rd-order approximation in orange uses the samples and the first derivatives. One order more is achieved by additionally using the second derivative or curvature information, for which the result is shown as the magenta-coloured graph. Last, the linear information for the variables is used to form an approximation of third order for the variables. The violet graph shows the approximation resulting from applying the objective function to the approximation of the variables. As before, the green stars mark a high-resolution Pareto front which is considered to be the true Pareto front.

The results show that all approximations using sensitivity information approximate the Pareto front accurately in this situation. This similarity results in no differences in the quality of the approximations between the different approaches. However, the approach via variables does yield not only an approximation of the Pareto front but also approximations for each variable.

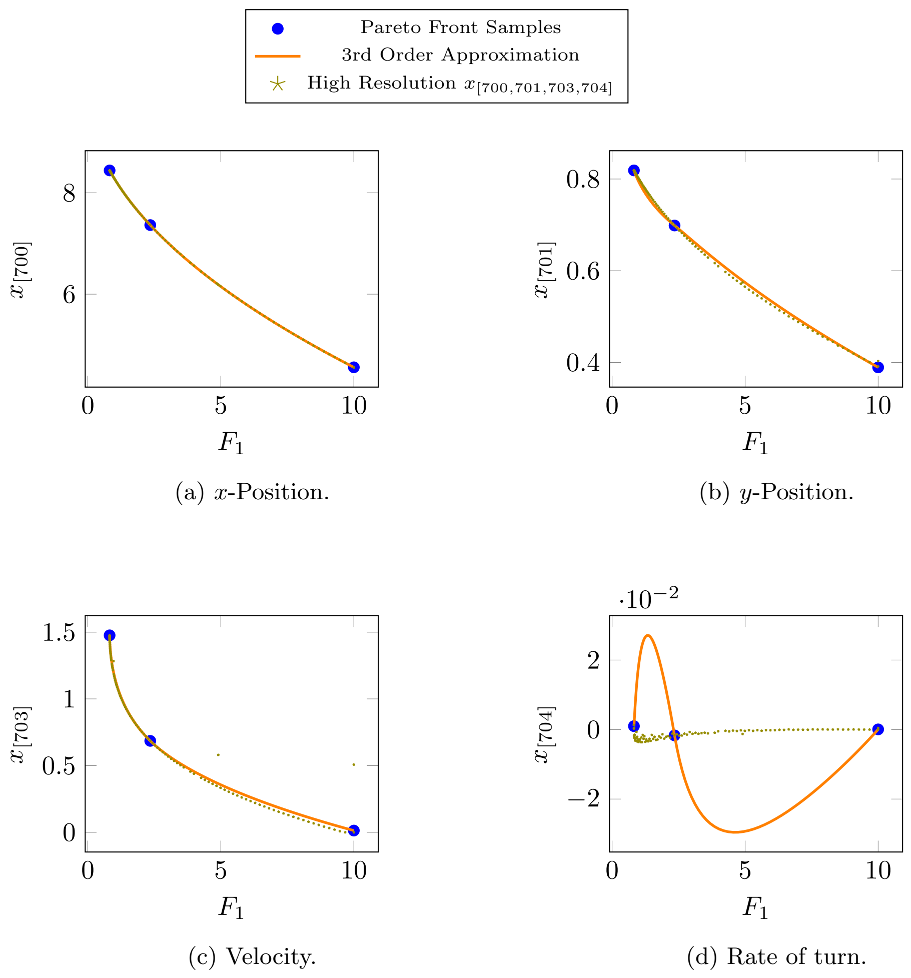
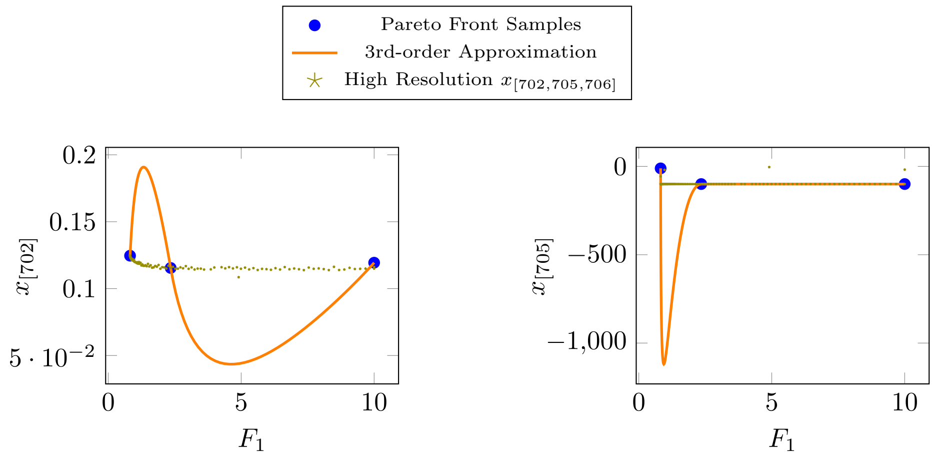


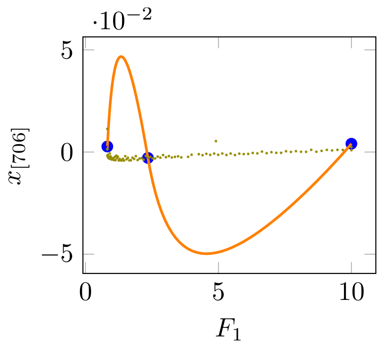
Figure 5.14: Continuous global approximation of the states in the final discrete point $\chi^{[N_t]}$ based on variables parametric sensitivity information of 3 samples for Example 5.3 with scaled objective functions (5.6).

Figures 5.14a to 5.14d show the global approximations for the variables on which the objective function explicitly depends. As expected, the global approximation for the rate of turn in Figure 5.14d does not approximate the rate of turn correctly. This is due to the already incorrect local approximation. The investigation of Figures 5.11d and 5.12 shows why this is the case and that the sensitivities are correct but only apply to a very local area. The fact that the overall approximation of the individual objective function and thus the Pareto front is still almost correct is due to the much lower scale of the rate of turn compared to the other parts of the objective function.

For the other three variables, namely x-position, y-position and velocity in Figures 5.14a to 5.14c the global approximations are accurate. How this result impacts the usability of the global variable approximation regarding the investigated example and situation depends on the intended use.
One possibility is that the decision-maker wants to get an overview of how the trajectories would look in the x-y-plane between individual Pareto front samples. In this case, the global approximation would provide a continuous view of the possible trajectories. It is also possible to compute the velocity for the discrete trajectory points. However, if the intention is to use the approximated trajectories for controlling the ship, the controller usually needs the ship's full state. This necessity might be problematic in this case due to the small validity range of the parametric sensitivities.



(a) Yaw. (b) EOT.



(c) Rudder angle.

Figure 5.15: Continuous global approximation of the yaw, EOT and rudder angle in the final discrete point $\chi^{[N_t]}$ based on variables parametric sensitivity information of 3 samples for the Example 5.3 with scaled objective functions (5.6).

The full state is even more problematic since the remaining states are also approximated

incorrectly. Figures 5.15a to 5.15c show the approximations for yaw, EOT and the rudder angle. The yaw and the rudder angle in Figures 5.15a and 5.15c show the same problematic behaviour as the rate of turn. This is not surprising, since these states are strongly coupled. However, the approximation for the EOT is also incorrect between the first and second sample. As in Section 3.3, this is due to the bang-bang nature of the EOT.
All in all, it is not recommendable in this case to use the global approximation of the whole state for this specific problem formulation. However, this approximation can be used to get an overview of how the trajectories develop in the x-y-plane, which is very helpful within the process of designing the control module. Note also that the incorrect global approximations provide one more piece of information, namely that the sensitivity information is probably only applicable in a close area around the sample. To discover this through sampling, the Pareto front would need significantly more samples and computation time. However, the global approximation reveals this problem at first glance, and in real-world examples, finding fundamental problems early in the development process is often most valuable.

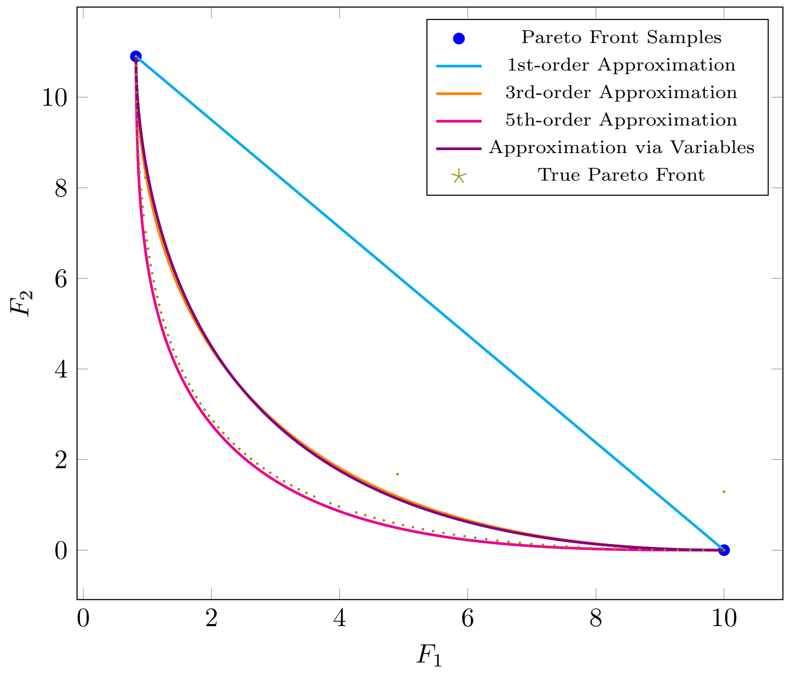


Figure 5.16: Continuous global approximation based on parametric sensitivity information of 2 samples for the Example 5.3 with scaled objective functions (5.6).

The overall goal in this section is to reduce the needed samples as far as possible. When using the global approximation techniques of the Pareto front, the lowest possible amount of samples is two. Figure 5.16 shows the resulting approximations for the example when created by the information of only individual minima and its parametric sensitivities. Comparing the linear approximation in blue, which is created without sensitivity information, and the other approximations, it is obvious that with the sensitivity information, it is immediately clear what the shape of the Pareto front is. This general knowledge about the shape was already information that the decision-maker could guess from the local approximations but should be emphasized here again since it is valuable information. This single piece of information makes it easier, for

example, to choose the scalarisation technique. Since the weighted sum is theoretically only able to sample the convex part of the Pareto front, the graphs in Figure 5.16 show clearly that this is the case here and that the weighted sum is applicable in case it should be chosen for its simplicity.
It is also impressive that the 5th-order approximation in magenta almost approximates the true Pareto front exactly. This good approximation clearly shows the potential of the proposed method. However, in a real-world application, the true Pareto front is unknown, and sampling it with high resolution would defeat the purpose of the proposed methods. Thus, it is hard to say if the decision-maker would trust in the approximation when looking at the real applicability. Mainly since the 5th-order approximation and the other two differ from each other while simultaneous approximation of 3rd-order in objective space and via the variables are almost the same. These differences could lead to the assumption that the latter two are correct and the 5th-order approximation is not. In this situation, it is advisable to compute the third sample in the middle of the other two, which would again lead to the results in Figure 5.13. The decision-maker can see that the 5th-order approximation does not change between both versions and that all three sensitivity based approximations are almost the same. Both information hint at the fact that the approximations are correct. In this way, it is possible with only three samples to create a trustworthy and accurate view of the overall Pareto front and guide the controller development with valuable information.

5.3 Conclusion of the Applicability of Multi-Objective Optimisation to Trajectory Optimisation Problems

This chapter shows how the investigated methods can be applied to a trajectory optimisation problem from the context of nMPC control of autonomous ships. The goal is to evaluate how different methods perform on these problems and emphasise the possibilities of multi-objective optimisation and the proposed methods within the development process of a model predictive control module for autonomous ships.
First, the application of multi-objective optimisation in the development of a controller for autonomous ships is motivated, afterwards, the fundamentals of the investigated example are explained in Section 5.1.

In Section 5.2.1 the MOSQP, NsSQP, weighted sum and Pascoletti-Serafini methods are applied to find a discrete approximation of the Pareto front. The comparison shows that all four methods can give a good overview of the general shape of the Pareto front. Differences are in the distribution of the sample points, where the Pascoletti-Serafini scalarisation can compute the best-distributed samples in terms of equidistant samples. In contrast, the weighted sum shows a high concentration of samples towards one of the individual minima.
The MOSQP and NsSQP algorithms are also able to find a sufficiently well distributed approximation. In the example presented, there is only a slight difference between the MOSQP and NsSQP. Thus the advancement made in Chapter 4 which does show an improvement in robustness, does not make an essential difference in this particular example. It is discussed that this is probably because the Pareto front in the investigated example is convex and does not show the critical properties which were discussed in Section 4.2.
As another difference, the section discusses that MOSQP and NsSQP do not depend as heavily on hyperparameters as the weighted sum and Pascoletti-Serafini approaches to sample the whole Pareto front. Thus they are suited if the hypothetical decision-maker wants to get a quick overview without much preparation. In contrast, the weighted sum and Pascoletti-Serafini meth-

ods are better suited if the approximation is refined after the first approximation since this only involves constraining the hyperparameter.
It is also emphasised how scaling the objective function influences the distribution of all computed approximations. Last it is discussed how the approaches perform if the number of samples is reduced.

Section 5.2.2 discusses how continuous approximations of the Pareto front based on parametric sensitivities can be utilized. These techniques have been introduced in Chapter 3 and in this chapter they are applied to the autonomous ship example. First, it is discussed how local approximations help get an overview of the Pareto front and how they enable the decision-maker to make educated decisions with very few samples. It is shown how the different types of parametric sensitivities can be used and which differences exist. The section also emphasises some pitfalls when using parametric sensitivities, mainly due to the information's locality and the requirements to make them usable.
Afterwards, the section investigates globally continuous approximations of the Pareto front. The section discusses how accurate the approximations approximate the Pareto front and how this can be used in the development process of an nMPC module or in the module itself. As before, the differences between the usage of different types of parametric sensitivities are discussed. It is also shown that the sensitivities for the variables might be problematic to use for a global approximation but can at the same time help to discover problematic behaviour of the solutions early in the development process.
All in all, the section shows how parametric sensitivity information can be utilised to quicken the development process, save computation time and provide deeper insights into the problem investigated. All of these features are most helpful when developing an nMPC module for real-world applications.

Chapter 6

Considerations for the Implementation of a General Multi-Objective Library for the NLP-Solver WORHP

Contents

The focus of this thesis was to improve existing and develop new algorithms to enhance the performance within multi-objective optimisation. The concrete implementation and especially the optimisation of the code itself was not the goal. However, during the development, some lessons were learned which provide general hints into how a general-purpose library could be implemented and how it would benefit from parallelisation.
This chapter provides some considerations on the implementation of a general-purpose multi-objective library for the solver WORHP. These considerations take into account the library's architectural design, especially regarding the application programming interface (API) for the possible user. In addition, some implementation details are discussed. Since multi-objective optimisation benefits heavily from parallel computation, the focus is on asynchronous computation and multi-threading.

6.1 A Unified Application Programming Interface for the NLP-Solver WORHP

This thesis does not only describe the algorithms on a theoretical basis but also proposes an add-on library for WORHP, which includes the investigated algorithms. This add-on is able to deal with multi-objective nonlinear problems (MONLP) as well as with multi-objective optimal control problems (MOOCP). The latter is not discussed in this thesis, but the library provides a way to discretise the MOOCP similar to the way TransWORHP [74] does. In addition, the library will provide interfaces to solve single-objective (NLP) and optimal control problems. The goal is to provide a unified application programming interface (API) for WORHP where the user needs to change as little as possible to switch between solution methods or problem types. The proposed library also focuses on separating the preprocessing modules, such as the scalarisation of MONLPs or the discretisation of OCPs from the actual WORHP module. In this way, it is possible to slide in other strategies in between the modules. These strategies could, for example, be parallelisation approaches as they are described in [50] or in more detail in [49].

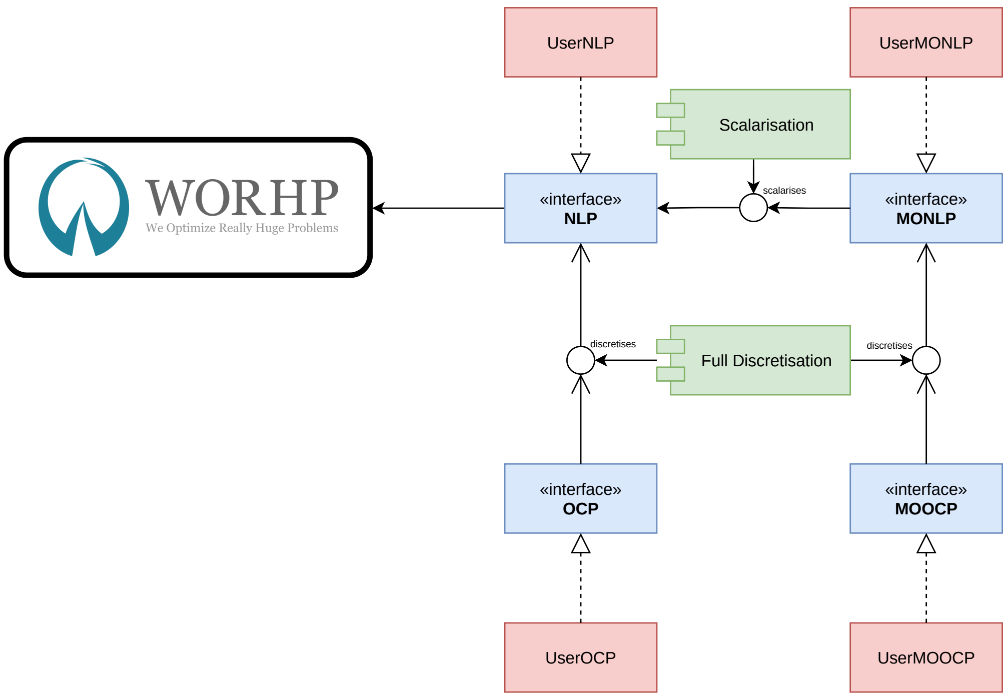


Figure 6.1: Domain model of the API for WORHP.

In Figure 6.1 the so-called domain model (see Evans [34]) of the most important modules is presented. It displays in red the four possibilities of a user to define an optimisation problem. These user-provided definitions need to implement the interfaces of the various types of problems. Those interfaces are displayed in blue. The green modules operate on these interfaces and transform the user input step by step into a problem that is solvable by a NLP-Solver such as WORHP (see Section 2.3).
The advantage of the design shown in Figure 6.1 is that the discretisation performed for the optimal control problems produces (NLP) or (MONLP) but that those classes do not depend on the discretisation in any way. Thus it is effortless to introduce new algorithms or strategies for solving (MONLP) such as parallelisation of the methods. Also, since the same module discretises OCP and MOOCP, both benefit directly from new methods such as different integration schemes.

6.2 Implementation of the NsSQP Algorithm

Section 4.3 shows how the implementation of the NsSQP Algorithm can benefit from using WORHP within the iterations. The following section will provide some more details about the implementation using WORHP. Thereafter, a discussion on possible ways for parallelisation is given.

6.2.1 Utilisation of WORHP for the Individual Phases of the NsSQP Algorithm

The details on how WORHP is utilised in the NsSQP algorithm differ between each individual phase of the algorithm. This section provides some more details on the implementation for each phase individually.
First, a brief reminder of the individual phases of the NsSQP algorithm is given here, the details are described in Algorithm 4.13.

Initialisation Phase The initial set of Points $X^{[0]} \subset \mathbb{R}^{N_x}$ is chosen with $0 < \bar{n} \in \mathbb{N}$ as the number of points. Several different initialisation strategies are described in Section 4.5.

Spread Phase The points in $X^{[k]}$ are spread by doing one SQP step for each point in the direction of each objective. Afterwards a cleanup strategy described in Section 4.2 is applied.

Refinement Stage The points in the final $X^{[k]}$ taken from the spread stage are refined towards Pareto-optimality, using a sum of all objectives and additional constraints to preserve the spreading.

The initialisation phase is straightforward in its implementation. If solving any problems of type (NLP) is necessary, they are solved with WORHP using the interface for single-objective problems. It is possible to provide a different WORHP parameter setting for this stage. This possibility enables the user, for example, to weaken tolerances and thus to save time in this phase. However, such a setting would be up to the user and is not part of the implementation.

Using WORHP for Single Steps within the Spreading Stage of the NsSQP Algorithm

In the spreading stage of the NsSQP algorithms, every iteration performs for all points in $X^{[k]}$ one QP step in the direction of each objective. However, the design of WORHP is not directly suited for a “one-step” mode. Thus the proposed implementation provides WORHP with the points from $X^{[k]}$ as an initial guess. Note, however, the variables x and the Lagrange multipliers λ and μ must be provided as an initial guess to work as expected.

Note that the termination criterion of WORHP must be set to perform one iteration at maximum. This setting will usually result in an unsuccessful optimisation flag from WORHP after this steps because the maximum number of iterations is reached. Thus the NsSQP implementation needs to differentiate between the flag for the maximum number of iterations reached and other unsuccessful termination criteria such as “minimum stepsize reached in Armijo” (see [117] for details).
In addition, not all parameter settings for WORHP are suited for this kind of exploitation. For example, the Filter Method [42, 72, 117] is problematic since it uses “historical” data from prior iterations. These iterations, however, were probably done for a different objective, thus rendering the information useless. The same holds for the BFGS method [15, 40, 57, 110]. In general, the settings for the spread phase need to be taken more carefully than usual when using WORHP.

To reduce the overhead of reinitialising WORHP before each optimisation, the restart technique of WORHP is used. This technique reuses all information from the former optimisation.

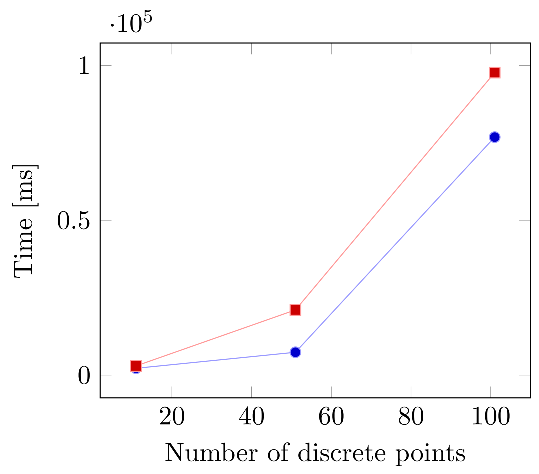


Figure 6.2: Benchmark results for the restart-technique of WORHP. Red: Timing without restart. Blue: Timing with restart.

The user only has to set the changed values. However, in this case, the initial guess has to be set anew, which reduces the restart to structure information about the derivative matrices and the parameters. Still, it reduces the computational effort. Especially the initialisation of used WORHP parameters should not be done by reading them every time from a file, because this would include the reading of information from the user's hard drive, which is generally very slow. Also, the WORHP internal memory is reused, which saves time for reallocating memory. However, using the restart technique makes it harder to parallelise the code. This problem is treated later in Section 6.2.2. First, the benefit of this technique is benchmarked at this point.

In order to benchmark the restart technique, an NLP was considered, which results from the discretisation of an OCP modelling the control of a car by a straightforward single-track model (see Example 3.3). The model consists of 2 states and 1 control, resulting in 3 optimisation variables and 2 constraint equations per discrete point.
Thus the initialisation of structures within WORHP becomes more complex with a higher number of discrete points. Figure 6.2 shows the time needed for varying numbers of discrete points. The times are measured as follows: Initially, the problem is solved once. Then the time needed for solving the problem again for 100 times with and without restart is measured. Only the structures were reused and not the resulting optimal variables x^*.
It can be seen that the blue line, representing the results with restart, runs below the red line, which represents the timings without a restart. Thus it becomes clear that using the restart technique will result in a better performance of the NsSQP algorithm. However, the exact benefit is hard to determine beforehand since it depends on how long the initialisation process takes compared to the actual solving. This distribution, in turn, depends strongly on the problem and parameter settings. Both times grow with the problem size.

Note 6.1
It should be noted that the usage of the restart technique described above does not represent a typical usage of the restart technique. Thus the timing shown here should be considered as very specific to the usage within the NsSQP algorithm.
For example in nMPC applications where very similar problems are solved in a high frequent

loop, the optimal variables x^* of one problem are taken as initial guesses to the next. Here, the restart is expected to provide even more efficiency.

Using WORHP for Iteration within the Refining Stage of the NsSQP Algorithm

Within the optimality-refining stage the NsSQP algorithm introduced in [46] iteratively performs one single QP step for each point in $X^{[k]}$ based on Problem (4.33). However, since the set $X^{[k]}$ does not grow in this stage, there is no cleanup necessary in-between the iterations. Additionally, in contrast to the original implementation, all regularisations and stepsize strategies within this thesis are done within WORHP. Thus it is not necessary to give the control back to the main NsSQP algorithm between the iterations. In order to prevent the overhead of switching between WORHP and the NsSQP Algorithm itself, it is proposed here to solve problem (4.33) for each point in $X^{[k]}$ until WORHP terminates.

As proposed in Note 4.9 it is sometimes useful to introduce a recovery strategy for points that were infeasible to solve due to the introduction of additional constraints in (4.33). The proposed strategy is to find a feasible point near to the original point $x^{[k]}$. This strategy can be realised by using the "feasible only" mode of WORHP, which neglects all influences of the objective function within the optimisation. Through this WORHP finds a (typically not unique) solution to the equation system introduced by the constraints $g(x)$ and $h(x)$ in (4.31) which can then be used as $x^{[k+1]}$. There is no guarantee that $\left\|x^{[k]} - x^{[k+1]}\right\|$ will be small by any means. However, applying this strategy might result in a better approximation of the Pareto front since it does not change any of the already successfully converged points and might introduce new ones.

6.2.2 Parallelisation Strategies for the NsSQP Algorithm

The NsSQP algorithm offers in its structure several possibilities for parallel computation. At this point, the individual parallelisation possibilities for the respective stages of initialisation, spread and refinement, are briefly introduced. Thereafter, it is explained which approach was chosen for the implementation. Finally, a short outlook to the further refinement of the parallelisation is given.

For the initialisation phase, the greatest potential for parallel calculations lies in the creation of the initial point set $X^{[0]} \subset \mathbb{R}^{N_x}$. Here, however, the possible advantage strongly depends on the chosen strategy. The Strategies 4.16, 4.18 and 4.21 do not include heavy computations. While it is possible to calculate every loop iteration within these algorithms in parallel, the benefit would likely be negligible. For the Strategies 4.19, 4.22 and 4.5.7 which are based on the computation of individual minima or which include refinement computation, the possible benefit of parallelisation is higher. Therefore, it is advantageous here to regard all calls of WORHP for solving an optimisation problem as a single parallel task. For these reasons, the implementation defines one parallel task for each solution computation with WORHP and computes everything else sequentially.

Within the spread phase, the NsSQP algorithm can benefit greatly from parallel computation. In each iteration $\bar{n}N_F$ independent QP-optimisations are performed. Looking at Algorithm 4.13, it becomes clear that step (II).1. which includes the spreading of all points $x_\ell^{[k]} \in X^{[k]}$ in all directions as well as step (II).1.(i) which spreads one single point $x_\ell^{[k]} \in X^{[k]}$ in the directions of all objectives, contain loops in which independent computations are performed. Hereby

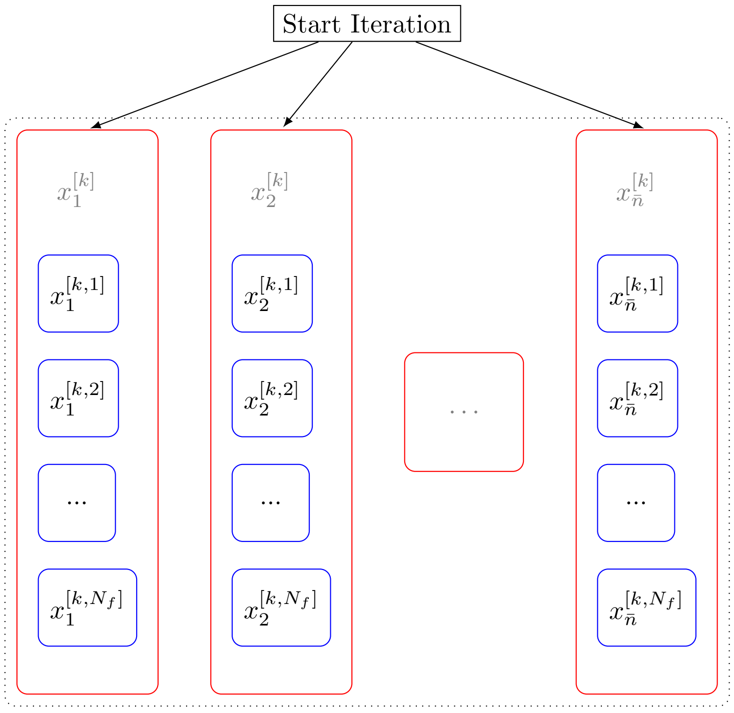


Figure 6.3: Parallelisation approach for spreading phase of the NsSQP algorithm using several problems per task. Red: Tasks containing all optimisation problems for one point in $X^{[k]}$.

(II).1.(i) is contained in step (II).1. Approaches for parallelisation can be to define one task for each iteration in either of these loops. However, if one parallelises (II).1.(i) the computational effort done in (II).1. outside the contained loop becomes negligible. Thus it seams reasonable to compute one or the other in a parallel but not both together.
The approach to define one task for each iteration in (II).1. is presented in Figure 6.3. The figure shows that the tasks (red) contain multiple optimisation problems (blue) that are computed sequentially. The advantage of choosing this approach is that it introduces less parallelisation overhead since it uses only $\bar{n}$ tasks instead of $\bar{n}N_F$. Due to the sequential calculation of multiple optimisation problems, it also can reuse all WORHP structures within one task and perform WORHP restarts on them with the original point $x_\ell^{[k]}$ as the initial guess. However, the disadvantage of this approach is that it can introduce quite large tasks (e.g. when N_F is larger). Since the end of each iteration introduces a synchronising point, this approach has a bigger risk that at the end of each iteration, one thread still has a heavy computation while all other threads are idle.

In the case that the number of cores is approximately $\bar{n}$ or even larger than $\bar{n}$ the approach described above would waste potential for parallelisation because it would have idle cores all the time. Therefore, it is advantageous in this situation to split an iteration of the spread phase into more individual tasks. This can be done by following the approach to parallelise the loop in (II).1.(i) and introduce one task for every optimisation problem as is shown in Figure 6.4. Here each problem (red) defines one task. The problems are then solved in parallel by worker threads, which produce the corresponding solutions (green). It introduces a bigger overhead but also will have less idle threads at the end of an iteration. However, in straightforward implementation the

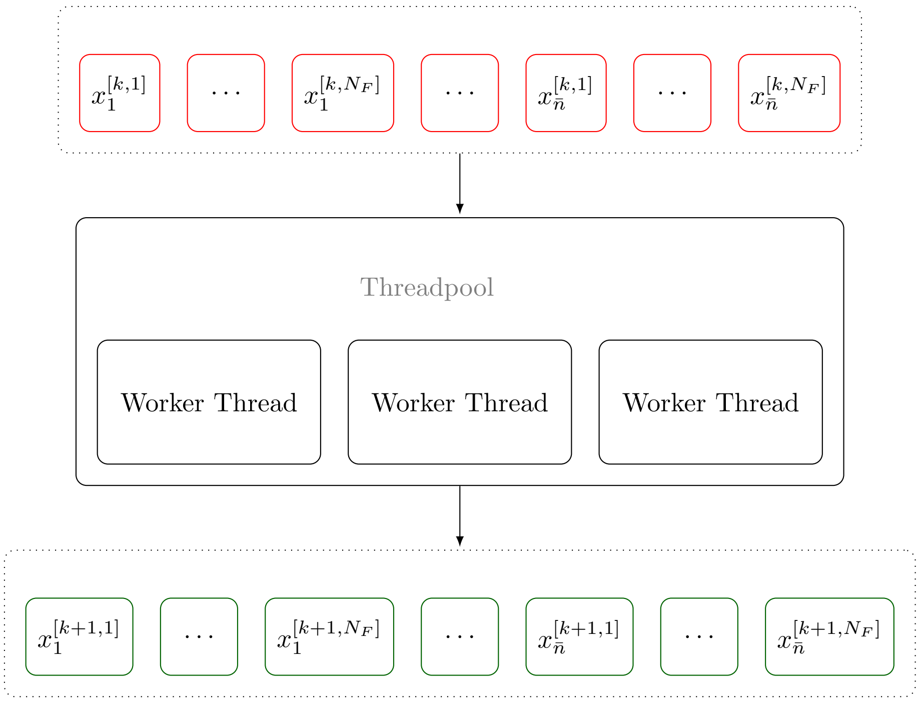


Figure 6.4: Parallelisation approach for spreading phase of the NsSQP algorithm using one problem per task. Red: Tasks containing one optimisation problem each. Green: Corresponding solutions.

advantage of reusing initialised WORHP structures would be gone because the tasks would not be able to share the WORHP instances.

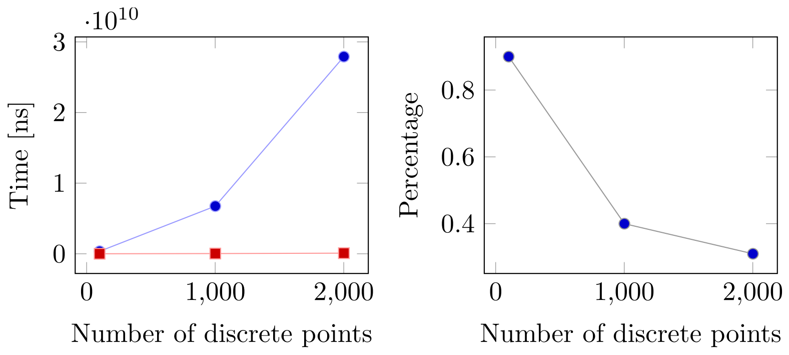


Figure 6.5: Benchmark internal processes of WORHP solution. Left: Blue: Time needed for solving one problem, Red: Time needed for initialising the constraints Jacobian structure. Right: Blue: Percentage share that the initialisation of the structures has in the total time.

The potential benefit of using WORHP restart is shown in Figure 6.5. On the left side, the blue line shows how much time the computation of one solution takes, whereas the red line shows

how much time the initialisation of sparsity structures of the constraints Jacobian takes. For the benchmark Problem 3.3 was used. Thus the number of discrete points directly correspond with the number of variables and constraints. Therefore, the dimension of the constraints Jacobian grows also with the number of discrete points. It can be seen here that the time needed for initialising the structures is, compared to the overall time, close to zero. The percentage of the overall solution time needed for structure initialisation also supports this observation. The resulting numbers are plotted on the right side of Figure 6.5. The numbers directly show that initialisation does take less than 1% of the overall time and drops down below 0.4% with increasing problem size. This decrease results from the higher number of iterations needed to solve a single problem when the dimension grows.
It is important to remember that within the spreading phase of the NsSQP algorithm, each “solution” is only one single SQP iteration. Nevertheless, with a straightforward implementation, the structures would be reinitialised for every single problem. Thus it is more of an interest here how much more time the initialisation takes compared to one single iteration.

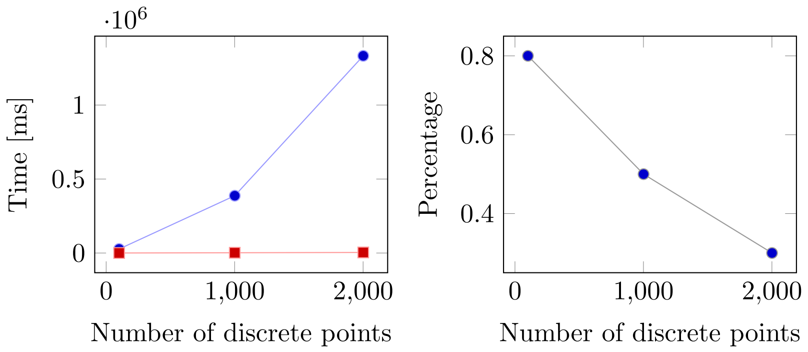


Figure 6.6: Benchmark time needed for initialisation of structures within spread stage. Left: Blue: Time needed for spread phase with 5 iterations, Red: Time needed for initialising the constraints Jacobian structures. Right: Blue: Percentage share that the initialisation of the structures has in the total time.

Figure 6.6 shows the timings for the OCP benchmark problem within the NsSQP spread phase. It is clearly visible that the numbers do not differ much from the numbers in Figure 6.5. Thus, it can be stated that for this single problem the initialisation of WORHP structures does not require more than 1% within the NsSQP spread phase. However, it also shows that the time which the initialisation takes within the whole solving problem, expressed in percent, decreases with the number of discrete points and thus with an increasing problem dimension. This hints in the direction that the complexity of the problem, which also grows with the dimension, overshadows the effect of the initialisation.

The time measurements for the whole test set are plotted in Figure 6.7. For the test set the initialisation takes between approximately 0% and 8% percent of the whole solving process. Interestingly it seems that the problems with only box constraints that are shown in red takes a significantly greater amount of time for the initialisation of WORHP structures. One possible reason for this could be that the corresponding QPs are easier to solve for WORHP which leads to less computational effort for the solving itself. However, the problems shown in red do not need to define any derivative structures. Thus it would be a reasonable assumption that the allocation of memory already needs enough time to be measurable.

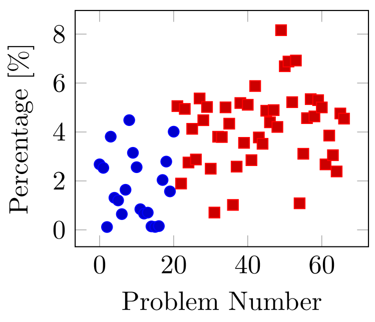


Figure 6.7: Percentage of computation time needed for initialisation of structures within spread stage with 20 iterations. Blue: Problems with nonlinear constraints, Red: Problems with only box constraints.

All in all it can be stated that the effect of using WORHP restart does seem beneficial for this algorithm. Therefore it is advantageous to develop a strategy to combine the restart technique with the highest possible parallelisation.

For this, it is important to remember, that for the NsSQP algorithm the restart only reuses the problem structures like the sparsity information of the derivative matrices. The initial variables and Lagrange multipliers are always set for each optimisation based on the values from the point that is currently spread. With this in mind it becomes clear, that the possibility of using a restart does not depend on sequentially programming all the spreading problems for one point. Rather it can be said, that the structure of the constraint Jacobian and Hessian of the Lagrangian does not change during the spread phase at all. The structure of the objective gradient might change, however. Thus WORHP structures could be reused for all optimisation problems if the objective gradient structure is reinitialised with the initial guesses for each optimisation. Especially the dimensions N_F, N_x, N_g, N_h do not change within the spread phase, thus using WORHP restart would at least save several memory allocations and memory frees.

In order to make use of all the advantages mentioned above, the proposed implementation changes the threadpool itself. Since the WORHP structures should be reused for every optimisation, they can be moved directly into the worker threads. In this way it is possible to reuse them, even if the problems are solved in no specific order. The resulting implementation pattern is shown in Figure 6.8. The threads within the threadpool contain the WORHP structures and will take one problem after another until all problems are solved. Hereby they will reuse the initialised WORHP structures.

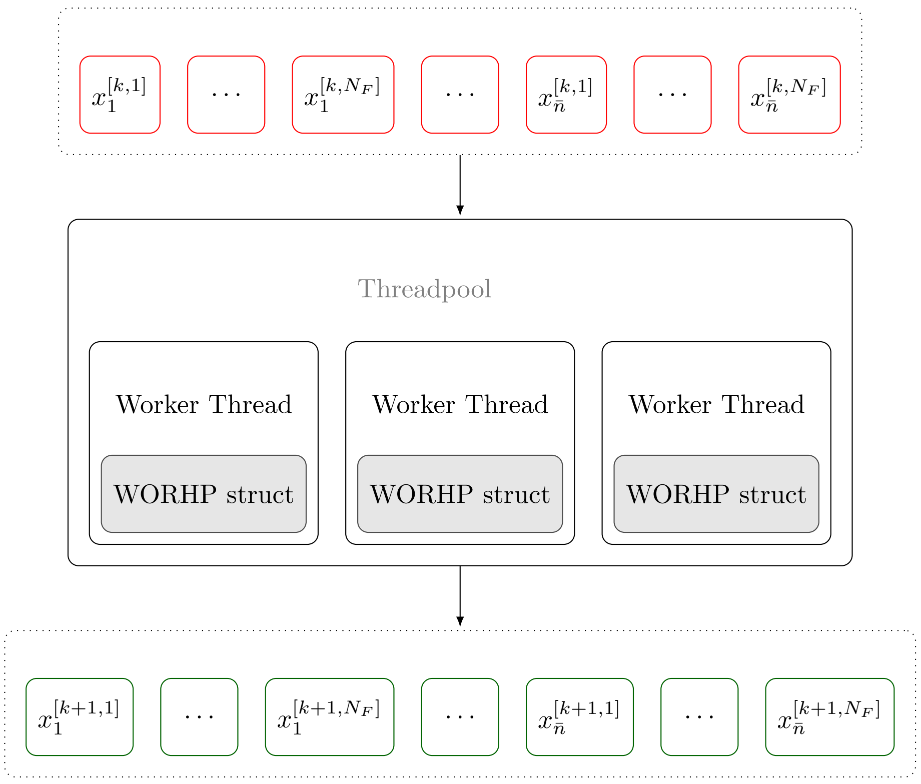


Figure 6.8: Parallelisation approach using WORHP structures as resources of the threads in a threadpool. Red: Tasks containing one optimisation problem each. Green: Corresponding solutions.

Chapter 7

Conclusion and Outlook

Contents

The Chapters 3, 4 and 5 already give concluding discussion on their respective results. This chapter briefly recalls these results and provides an additional conclusion with the focus on the overall goals given in Chapter 1. Afterwards an outlook to possible future developments is given.

7.1 Conclusion

The goal of this thesis was to improve multi-objective optimisation tools. First, the fundamentals of multi-objective optimisation and the scalarisation method from Pascoletti and Serafini were described. Additionally, the fundamentals of parametric sensitivity analysis were introduced. Both the Pascoletti-Serafini method and parametric sensitivity analysis were combined such that it was possible to compute the sensitivity of the individual objectives for the hyperparameter in the scalarisation method. This combination led to the idea of building continuous approximations of the mapping from the hyperparameter onto the Pareto front, thus extending existing ideas.
The continuous approximations were first computed separately around each sample. Besides the already known interpretation as the tradeoff between individual objectives, these approximations led to a better overlook of the Pareto front's shape. The introduction of second-order approximations based on the parametric sensitivities of the Lagrange multiplier improved the results further. It was possible to get an accurate impression of the problem's structure with a highly reduced number of samples in academic examples. These results were further extended by applying the sensitivities of the variables. This application made it possible to approximate the mapping from the hyperparameter into the variable space. Based on this approximated mapping, another approximation of the Pareto front becomes possible. This additional approximation showed a robust behaviour on one academic example.
Besides the local approximation at each sample, a continuous approximation over the whole Pareto front was created. Based on the first- and second-order derivatives in objective space and variable space, polynomial interpolations of third and fifth degree were created for a bi-objective example. The potential to reduce the samples needed to approximate the Pareto front was shown as well as some problems when using the local information of the parametric sensitivity analysis for a global approximation.
Some limitations of the proposed approximation technique were discussed. These problems originated in the local nature of the information from parametric sensitivity analysis. It was also

discussed how the methods could be generalised for more than two objectives. For the local approximation, it was shown that it is directly possible to extend the technique to higher dimensions.
Overall, the results for approximations based on parametric sensitivity analysis showed a high potential of the method to reduce the samples needed to compute an accurate approximation of the Pareto front. The additional information from several different approximations also improves the robustness and provides a heuristic measure on the quality of the approximations.

For the second improvement on multi-objective tools, the MOSQP algorithm was discussed. After the fundamentals of the algorithm were introduced, it was shown how the cleanup strategy influences the computed Pareto front approximation. The cleanup strategy keeps an intermediate set of points in a reasonable size throughout the spread phase of the algorithm. It was proposed to reuse the strategy of nondominated-sorting from the heuristic NSGA-II algorithm thus leading to the NsSQP algorithm. Also, new strategies to create the initial set of points for the MOSQP algorithm were described. Those new strategies were mainly based on convex combinations between individual optima of the respective problem.
Performance profiles, based on purity-, spread- and hypervolume metrics as well as data profiles, did show that the proposed changes improve the algorithm's overall performance on a given set of test problems. Especially regarding the spread metrics, a better result than before could be accomplished. Compared to other algorithms like the NSGA-II and the weighted sum, the new NsSQP algorithm did show a good performance. Especially the purity performance profile and data profiles showed a good result compared to the NSGA-II algorithm. However, these results were similar to the original results for the MOSQP algorithm. Thus it was shown that the proposed changes did improve the algorithm but did not significantly impact general advantages and disadvantages compared to other algorithms. It was also discussed how the algorithm could be implemented to utilise modern parallel hardware architecture. This parallelisation significantly reduced the computation time of the algorithm.

The goal of the improvements made in this thesis was to improve the development process for automated systems. This goal was addressed by applying the investigated methods and their improvements on an example of autonomous ship control from the project GALILEOnautic 2. After the example was described, the Pascoletti-Serafini and weighted sum scalarisation methods, as well as the NsSQP algorithm, were applied to the example. This application led to different discrete approximations of the Pareto front that provided insight into the tradeoffs between the two investigated objectives. It was shown which information these discrete approximations provide and how the differences between the methods influence the quality.
Subsequently, the continuous approximation techniques based on parametric sensitivity analysis were applied. The results showed that these techniques reduced the number of samples needed for a good approximation, thus fastening the development process. Also, possible problems with the techniques related to the local nature of the parametric sensitivity information were shown. Especially the sensitivities of some of the variables showed problems in the investigated example. These problems were mainly due to variables corresponding to bang-bang controls but also due to high frequent oscillations in the variables over the Pareto front. For the latter, it was also discussed how the parametric sensitivities help in revealing these high frequent structures besides the problems they cause. The results for this particular example problem also showed that the problems in globally approximating the variables do not necessarily need to show up in the Pareto front approximation based on the approximated variables.

Overall, the goal was achieved to improve the development process of automated systems by applying and improving tools from multi-objective optimisation. Especially the ideas based on parametric sensitivity analysis information show a high potential in this regard. The results showed that they improve the efficiency of the development process and provide new information that would be hard to get through other methods. The improvements in the MOSQP algorithm also help in the process when the algorithm is applied to get a general overview of the problems. Finally, the general idea of investigating optimal control problems in a nonlinear model predictive control module through multi-objective optimisation showed high potential to help the developer when several objectives were integrated.

7.2 Outlook

The continuous Pareto front approximation methods showed a good potential for reducing the number of samples needed for an accurate approximation on the investigated academic and real-world examples. Formal convergence results could strengthen the technique and help in better understanding the method. This result could also lead to an adaptive algorithm that uses an estimated approximation error to define intervals in which the next sample should be computed. Another idea is to use the second-order local approximation instead of the linear approximation in an adaptive Pareto front sampling algorithm. In this way, it would be possible to differentiate between two different types of distances between samples. The first would be the euclidean distance in objective space, which is already used for adaptive algorithms. The second would be the distance along the approximated Pareto front, which would be different since the approximation would then be a quadratic one. The sensitivities in variables space and the respective approximations could also be used similarly.

For the MOSQP, the impact of the chosen cleanup strategy was shown. It would be interesting to investigate other strategies here, especially if techniques based on machine learning could further improve the algorithm. In addition, changing the QP formulations in the spread phase to integrate different combinations of objectives and thus extend the possible search directions would be interesting. However, if this was done, the proof of convergence would need to be adapted for this idea.

In the application to optimal control problems, investigating the advantage of using multi-objective optimisation tools based on more real-world examples would be interesting. The expectation here is that the process of weight-tweaking in the integration of multiple objectives would be improved a lot. Here it would be interesting how an approach based on the Pascoletti-Serafini scalarisation would perform in a real-time system on a ship, car or other automated systems when compared to the weighted sum scalarisation.
Another idea is to investigate how multi-objective optimisation could help when changing hard constraints to soft constraints, thus introducing more objectives. When the problem is reformulated, parametric sensitivities on the new objective could be used to make statements that are not possible with hard constraints since they would then involve a change in the active set of the constraints.
Based on the information gained from the continuous approximations, automated decision-making modules could be developed. These modules could choose points from the continuous approximation to steer the system based on higher-level decision-making. The continuous approximations would reduce the number of samples needed but simultaneously allow the decision-making algorithm to choose from a continuous set of optimal compromises. In addition, the sensitivities for the variables could be used to apply a feasibility refinement for the chosen

points.
Also, the sensitivities themselves can be used in the higher-level decision-making algorithm. The idea would be to use the tradeoff information from the sensitivities to evaluate if a particular solution is favourable. For example, balanced tradeoffs could be one goal in this situation. Since the derivative of the tradeoffs is also available, it could be possible to build a derivative-based algorithm to find Pareto-optimal points with balanced tradeoffs automatically.

Bibliography

[1] R. Allmendinger, M. Ehrgott, X. Gandibleux, M. J. Geiger, K. Klamroth, and M. Luque. Navigation in multiobjective optimization methods. *Journal of Multi-Criteria Decision Analysis*, 24(1-2):57–70, 2017.

[2] W. Alt. *Nichtlineare Optimierung: Eine Einführung in Theorie, Verfahren und Anwendungen.* Springer-Verlag, 2013.

[3] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

[4] A. Berger. Code for dissertation. Mendeley Data, V1, doi: 10.17632/k4knrkpmxx.1, 2021.

[5] A. Berger and C. Büskens. Adaptive pareto front sampling based on parametric sensitivity analysis in a bi-objective setting. In *7th International Conference on Astrodynamics Tools and Techniques (ICATT)*, 2018.

[6] A. Berger, F. Jung, M. Echim, C. Büskens, and M. Schollmeyer. Optimal expansion planning for an electric distribution network with the nlp solver worhp. *PAMM*, 17(1):735–736, 2017.

[7] A. Berger, M. Knauer, and C. Büskens. Pareto front interpolation based on parametric sensitivity analysis in a bi-objective setting. *PAMM*, 18(1):e201800273, 2018.

[8] J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming.* SIAM, 2010.

[9] F. Biscani and D. Izzo. A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, 5(53):2338, 2020.

[10] J.-F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical optimization: theoretical and practical aspects.* Springer Science & Business Media, 2006.

[11] J. M. Borwein. On the existence of pareto efficient points. *Mathematics of Operations Research*, 8(1):64–73, 1983.

[12] S. P. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[13] L. Bradstreet. *The hypervolume indicator for multi-objective optimisation: calculation and use.* University of Western Australia Perth, 2011.

[14] J. Branke, K. Deb, K. Miettinen, and R. Słowiński. *Multiobjective optimization: Interactive and evolutionary approaches*, volume 5252. Springer Science & Business Media, 2008.

[15] C. Broyden. The convergence of a class of double-rank minimization algorithms. ii. the new algorithm. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90, 1970.

[16] C. Büskens. *Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen.* PhD thesis, Universität Münster, Münster, 1998.

[17] C. Büskens. Real-time solutions for perturbed optimal control problems by a mixed open- and closed-loop strategy. In *Online optimization of large scale systems*, pages 105–116. Springer, 2001.

[18] C. Büskens. Echtzeitoptimierung und Echtzeitoptimalsteuerung parametergestörter Probleme. *Universität Bayreuth, Fakultät für Mathematik und Physik, Habil.*, 2002.

[19] C. Büskens and D. Wassel. The ESA NLP solver WORHP. In G. Fasano and J. D. Pintér, editors, *Modeling and Optimization in Space Engineering*, volume 73, pages 85–110. Springer New York, 2013.

[20] V. Chankong and Y. Y. Haimes. *Multiobjective decision making: theory and methodology.* Courier Dover Publications, 2008.

[21] S. Chen, D. Wassel, and C. Büskens. High-precision modeling and optimization of cogeneration plants. *Energy Technology*, 4(1):177–186, 2016.

[22] N. Chopin. Fast simulation of truncated gaussian distributions. *Statistics and Computing*, 21(2):275–288, 2011.

[23] A. R. Conn, N. I. Gould, and P. L. Toint. *Trust region methods.* SIAM, 2000.

[24] A. L. Custódio, J. A. Madeira, A. I. F. Vaz, and L. N. Vicente. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21(3):1109–1140, 2011.

[25] I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural optimization*, 14(1):63–69, 1997.

[26] I. Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM journal on optimization*, 8(3):631–657, 1998.

[27] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

[28] M. Dellnitz, J. Eckstein, K. Flaßkamp, P. Friedel, C. Horenkamp, U. Köhler, S. Ober-Blöbaum, S. Peitz, and S. Tiemeyer. Development of an intelligent cruise control using optimal control methods. *Procedia Technology*, 15:285–294, 2014.

[29] S. Di Cairano and I. V. Kolmanovsky. Real-time optimization and model predictive control for aerospace and automotive applications. In *2018 annual American control conference (ACC)*, pages 2392–2409. IEEE, 2018.

[30] W. Domschke, A. Drexl, R. Klein, and A. Scholl. *Einführung in Operations Research.* Springer-Verlag, 2015.

[31] M. Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[32] G. Eichfelder. *Adaptive Scalarization Methods in Multiobjective Optimization.* Springer Verlag, 2008.

[33] G. Eichfelder. An adaptive scalarization method in multiobjective optimization. *SIAM Journal on Optimization*, 19(4):1694–1718, 2009.

[34] E. Evans and R. Szpoton. *Domain-driven design.* Helion, 2015.

[35] G. Farin. *Curves and surfaces for computer-aided geometric design: a practical guide.* Elsevier, 2014.

[36] A. V. Fiacco. Sensitivity analysis for nonlinear programming using penalty methods. *Mathematical programming*, 10(1):287–311, 1976.

[37] A. V. Fiacco. Nonlinear programming sensitivity analysis results using strong second order assumptions. Technical report, George Washington University Washington DC Inst. for Management Science and Engineering, 1978.

[38] A. V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming.* Academic Press, 1983.

[39] A. V. Fiacco and Y. Ishizuka. Sensitivity and stability analysis for nonlinear programming. *Annals of Operations Research*, 27(1):215–235, 1990.

[40] R. Fletcher. A new approach to variable metric algorithms. *The computer journal*, 13(3):317–322, 1970.

[41] R. Fletcher. *Practical methods of optimization.* John Wiley & Sons, 2013.

[42] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical programming*, 91(2):239–269, 2002.

[43] J. Fliege. Gap-free computation of pareto-points by quadratic scalarizations. *Mathematical Methods of Operations Research*, 59(1):69–89, 2004.

[44] J. Fliege, L. G. Drummond, and B. F. Svaiter. Newton's method for multiobjective optimization. *SIAM Journal on Optimization*, 20(2):602–626, 2009.

[45] J. Fliege and B. F. Svaiter. Steepest descent methods for multicriteria optimization. *Mathematical methods of operations research*, 51(3):479–494, 2000.

[46] J. Fliege and A. I. F. Vaz. A method for constrained multiobjective optimization based on SQP techniques. *SIAM Journal on Optimization*, 26(4):2091–2119, 2016.

[47] A. Flores-Tlacuahuac, P. Morales, and M. Rivera-Toledo. Multiobjective nonlinear model predictive control of a class of chemical reactors. *Industrial & Engineering Chemistry Research*, 51(17):5891–5899, 2012.

[48] O. Forster. *Analysis 1: Differential- und Integralrechnung einer Veränderlichen.* Springer-Verlag, 2012.

[49] S. Geffken. *Effizienzsteigerung numerischer Verfahren der nichtlinearen Optimierung.* PhD thesis, University of Bremen, Bremen, 2017.

[50] S. Geffken and C. Büskens. WORHP multi-core interface, parallelisation approaches for an NLP solver. In *6th International Conference on Astrodynamics Tools and Techniques (ICATT)*. ESA, 2016.

[51] S. Geffken and C. Büskens. Feasibility refinement in sequential quadratic programming using parametric sensitivity analysis. *Optimization Methods and Software*, 32(4):754–769, 2017.

[52] J. Gehrt, R. Zweigel, S. Roy, C. Büskens, M. Kurowski, T. Jeinsch, A. Schubert, M. Gluch, O. Simanski, E. Pairet-Garcia, et al. Optimal maneuvering and control of cooperative vessels within harbors. In *Journal of Physics: Conference Series*, volume 1357, page 012019. IOP Publishing, 2019.

[53] C. Geiger and C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Springer-Verlag, 2013.

[54] E. M. Gertz and S. J. Wright. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software (TOMS)*, 29(1):58–81, 2003.

[55] P. E. Gill and W. Murray. Numerically stable methods for quadratic programming. *Mathematical programming*, 14(1):349–372, 1978.

[56] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. SIAM, 2019.

[57] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26, 1970.

[58] J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

[59] S. Greco, J. Figueira, and M. Ehrgott. *Multiple criteria decision analysis*, volume 37. Springer, 2016.

[60] L. Grüne and J. Pannek. Nonlinear model predictive control. In *Nonlinear model predictive control*, pages 45–69. Springer, 2017.

[61] Y. Y. Haimes and V. Chankong. Kuhn-tucker multipliers as trade-offs in multiobjective decision-making analysis. *Automatica*, 15(1):59–72, 1979.

[62] R. Hartley. On cone-efficiency, cone-convexity and cone-compactness. *SIAM Journal on Applied Mathematics*, 34(2):211–222, 1978.

[63] M. Henig. Proper efficiency with respect to cones. *Journal of Optimization Theory and Applications*, 36(3):387–407, 1982.

[64] M. Henig. Value functions, domination cones and proper efficiency in multicriteria optimization. *Mathematical programming*, 46(1):205–217, 1990.

[65] C. I. Hernandez Castellanos, S. Ober-Blöbaum, and S. Peitz. Explicit multiobjective model predictive control for nonlinear systems under uncertainty. *International Journal of Robust and Nonlinear Control*, 30(17):7593–7618, 2020.

[66] C. Hillermeier et al. *Nonlinear multiobjective optimization: a generalized homotopy approach*, volume 135. Springer Science & Business Media, 2001.

[67] M. Jacobse and C. Büskens. Revisiting design aspects of a qp solver for worhp. 2018.

[68] J. Jahn et al. *Vector optimization.* Springer, 2009.

[69] F. Jung, C. Büskens, and M. Lachmann. Smartfarm–data based optimization for optimal energy management. *PAMM*, 17(1):741–742, 2017.

[70] P. Kalmbach. *Effiziente Ableitungsbestimmung bei hochdimensionaler nichtlinearer Optimierung.* PhD thesis, Universität Bremen, 2011.

[71] W. Karush. Minima of functions of several variables with inequalities as side conditions. In *Traces and Emergence of Nonlinear Programming*, pages 217–245. Springer, 2014.

[72] A. E. Kemper. Filtermethoden zur Bewertung der Suchrichtung in NLP-Verfahren, 2010.

[73] I. Y. Kim and O. L. De Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and multidisciplinary optimization*, 29(2):149–158, 2005.

[74] M. Knauer and C. Büskens. From WORHP to TransWORHP. In *Proceedings of the 5th international conference on astrodynamics tools and techniques*, 2012.

[75] M. Knauer and C. Büskens. Understanding concepts of optimization and optimal control with WORHP lab. In *Proc. 6th Int. Conf. Astrodynamics Tools Techniques*, 2016.

[76] M. Knauer and C. Büskens. Real-time optimal control using TransWORHP and WORHP zen. In *Modeling and Optimization in Space Engineering*, pages 211–232. Springer, 2019.

[77] F. Kohlmai. *Modellierung, Parameteridentifikation und optimale Drehzahlregelung eines Schiffsmotors im Gasbetrieb.* PhD thesis, Universität Bremen, 2020.

[78] R. Kuhlmann. *A primal-dual augmented lagrangian penalty-interior-point algorithm for nonlinear programming.* PhD thesis, Universität Bremen, 2018.

[79] R. Kuhlmann. Learning to steer nonlinear interior-point methods. *EURO Journal on Computational Optimization*, 7(4):381–419, 2019.

[80] R. Kuhlmann, S. Geffken, and C. Büskens. Worhp zen: Parametric sensitivity analysis for the nonlinear programming solver worhp. In *Operations Research Proceedings 2017*, pages 649–654. Springer, 2018.

[81] H. W. Kuhn. Nonlinear programming: a historical view. In *Traces and Emergence of Nonlinear Programming*, pages 393–414. Springer, 2014.

[82] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings 2nd Berkeley Symposium on Mathematical Statistics and Probability.* University of California Press, 1951.

[83] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer, 2014.

[84] M. Kurowski, S. Roy, J.-J. Gehrt, R. Damerius, C. Büskens, D. Abel, and T. Jeinsch. Multi-vehicle guidance, navigation and control towards autonomous ship maneuvering in confined waters. In *2019 18th European Control Conference (ECC)*, pages 2559–2564. IEEE, 2019.

[85] M. Lachmann, J. Maldonado, W. Bergmann, F. Jung, M. Weber, and C. Büskens. Self-learning data-based models as basis of a universally applicable energy management system. *Energies*, 13(8):2084, 2020.

[86] S. E. Li, Q. Guo, S. Xu, J. Duan, S. Li, C. Li, and K. Su. Performance enhanced predictive control for adaptive cruise control system considering road elevation information. *IEEE Transactions on Intelligent Vehicles*, 2(3):150–160, 2017.

[87] J. Majohr and M. Kurowski. Modellbildung und regelung unbemannter, autonomer oberflächenfahrzeuge (unmanned surface vehicles/usv). In *Maritime Regelungs-und Sensorsysteme*, pages 557–587. Springer, 2021.

[88] C. Meerpohl, K. Flaßkamp, and C. Büskens. Optimization strategies for real-time control of an autonomous melting probe. In *2018 Annual American Control Conference (ACC)*, pages 3756–3762. IEEE, 2018.

[89] H. F. Meyer. *Echtzeitoptimierung für Ausweichtrajektorien mittels der Sensitivitätsanalyse eines parametergestörten nichtlinearen Optimierungsproblems.* PhD thesis, Universität Bremen, 2016.

[90] E. Mezura-Montes and C. A. C. Coello. Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194, 2011.

[91] K. Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[92] P. Moin. *Fundamentals of engineering numerical analysis.* Cambridge University Press, 2010.

[93] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[94] T. Nikolayzik. *Korrekturverfahren zur numerischen Lösung nichtlinearer Optimierungsprobleme mittels Methoden der parametrischen Sensitivitätsanalyse.* PhD thesis, Universität Bremen, 2012.

[95] J. Nocedal and S. Wright. *Numerical optimization.* Springer Science & Business Media, 2006.

[96] D. Nowak and K.-H. Küfer. A ray tracing technique for the navigation on a non-convex pareto front. *arXiv preprint arXiv:2001.03634*, 2020.

[97] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[98] A. Pascoletti and P. Serafini. Scalarizing vector optimization problems. *Journal of Optimization Theory and Applications*, 42(4):499–524, 1984.

[99] S. Peitz. *Exploiting structure in multiobjective optimization and optimal control.* PhD thesis, Universität Paderborn, 2017.

[100] S. Peitz and M. Dellnitz. A survey of recent trends in multiobjective optimal control—surrogate models, feedback control and objective reduction. *Mathematical and Computational Applications*, 23(2):30, 2018.

[101] S. Peitz, K. Schäfer, S. Ober-Blöbaum, J. Eckstein, U. Köhler, and M. Dellnitz. A multiobjective mpc approach for autonomously driven electric vehicles. *IFAC-PapersOnLine*, 50(1):8674–8679, 2017.

[102] A. Probst, G. Peytavi, D. Nakath, A. Schattel, C. Rachuy, P. Lange, J. Clemens, M. Echim, V. Schwarting, A. Srinivas, et al. Kanaria: Identifying the challenges for cognitive autonomous navigation and guidance for missions to small planetary bodies. In *International Astronautical Congress (IAC)*, 2015.

[103] S. J. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. In *AIche symposium series*, volume 93, pages 232–256. New York, NY: American Institute of Chemical Engineers, 1971-c2002., 1997.

[104] S. Rauški. *Limited Memory BFGS method for Sparse and Large-Scale Nonlinear Optimization.* PhD thesis, Staats-und Universitätsbibliothek Bremen, 2014.

[105] M. J. Rentmeesters, W. K. Tsai, and K.-J. Lin. A theory of lexicographic multi-criteria optimization. In *Proceedings of ICECCS'96: 2nd IEEE International Conference on Engineering of Complex Computer Systems (held jointly with 6th CSESAW and 4th IEEE RTAW)*, pages 76–79. IEEE, 1996.

[106] M. Rick, J. Clemens, L. Sommer, A. Folkers, K. Schill, and C. Büskens. Autonomous driving based on nonlinear model predictive control and multi-sensor fusion. *IFAC-PapersOnLine*, 52(8):182–187, 2019.

[107] S. Roy, H. Wernsing, and C. Büskens. Optimization of ship manoeuvring within the project galileonautic. *PAMM*, 17(1):813–814, 2017.

[108] S. Schäffler, R. Schultz, and K. Weinzierl. Stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 114(1):209–222, 2002.

[109] A. S. Schattel. *Dynamic Modeling and Implementation of Trajectory Optimization, Sensitivity Analysis, and Optimal Control for Autonomous Deep Space Navigation.* PhD thesis, Universität Bremen, 2018.

[110] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

[111] L. Sommer, M. Rick, A. Folkers, and C. Büskens. Ao-car: transfer of space technology to autonomous driving with the use of worhp. In *Proceedings of the 7th International Conference on Astrodynamics Tools and Techniques*, volume 84, 2018.

[112] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[113] M. Stieler. *Performance Estimates for Scalar and Multiobjective Model Predictive Control Schemes.* PhD thesis, 2018.

[114] J. Thomann. *A trust region approach for multi-objective heterogeneous optimization.* PhD thesis, Technische Universität Illmenau, Illmenau, 2019.

[115] S. Ulbrich. On the superlinear local convergence of a filter-sqp method. *Mathematical Programming*, 100(1):217–245, 2004.

[116] Chair of Control Engineering, University of Rostock. https://www.rt.uni-rostock.de/en/arbeitsgruppe/.

[117] User Guide: We Optimize Really Huge Problems (WORHP). https://worhp.de/latest/download/user_manual.pdf. Accessed: 2020-05-17.

[118] A. I. F. Vaz and L. N. Vicente. A particle swarm pattern search method for bound constrained global optimization. *Journal of Global Optimization*, 39(2):197–219, 2007.

[119] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Local convergence. Technical report, Technical Report RC23033 (W0312-090), TJ Watson Research Center, Yorktown, 2003.

[120] D. Wassel. *Exploring novel designs of NLP solvers: Architecture and Implementation of WORHP*. PhD thesis, Universität Bremen, 2013.

[121] Website: Combinations of numbers sum up to 10. https://math.stackexchange.com/questions/1462099/number-of-possible-combinations-of-x-numbers-that-sum-to-y. Accessed: 2020-11-07.

[122] T. Weiskircher, Q. Wang, and B. Ayalew. Predictive guidance and control framework for (semi-) autonomous vehicles in public traffic. *IEEE Transactions on control systems technology*, 25(6):2034–2046, 2017.

[123] Website: Implementation of an Algorithm for Computing the Hypervolume of a Paretofrontapproximation. ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c. Accessed: 2020-07-25.

[124] M. Wiesner, A. Berger, W. Bergmann, K. Schäfer, C. Dittert, P. Schulpyakov, and C. Büskens. TOPAS Model Fitting: Improving the modeling process of dynamical systems using parametric sensitivities from nonlinear optimization [conference presentation]. International Conference on Operations Research, Bern, 2021. Abstract: https://www.euro-online.org/conf/admin/tmp/program-or2021.pdf.

[125] M. Wiesner, K. Schäfer, W. Bergmann, A. Berger, P. Schulpyakov, C. Dittert, and C. Büskens. Analyzing the influence of measurements in dynamical parameter identification using parametric sensitivities. manuscript submitted for publication. In *Proceedings of the Third IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON*. IFAC, 2021.

[126] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[127] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.