Optimization-Based Reachability Analysis for Landing Scenarios

Dissertation

submitted to University of Bremen
for the degree of Dr. rer. nat.

by

Kai Wah Chan

1st Reviewer:	Prof. Dr. Christof Büskens, University of Bremen
2nd Reviewer:	Dr. Stephan Theil, German Aerospace Center Bremen
Date of Defense:	October 11, 2022

Abstract

In this work, optimization-based algorithms are presented to approximate reachability sets. A reachability set comprises all dynamic system states that admissible controls can reach for a given initial state. Approaches based on the theory of optimization and optimal control have the advantage here that further constraints may be included in addition to the dynamics and the boundary conditions. However, solving nonlinear optimization tasks is considered costly in terms of time, so this should be done in a directed manner that yields as much insight as possible.
For this reason, the presented algorithms embed the necessary optimization runs in a geometric framework. In particular, convex sets and properties of polytopes are discussed in this work, which allows a structural notion of geometric objects in higher dimensions. The optimization theory is fundamental to this work in treating nonlinear and convex programs. In the dynamic context, optimal control processes are introduced, minimizing a cost functional subject to a first-order ordinary differential equation and other constraints. The parametric sensitivity analysis is an essential part of this work. It is a post-optimal tool to acquire additional knowledge from a solved optimization without significant effort.
Three algorithms for set approximation are presented. They are based on grids and polytopes. This work uncovers the parallelization potential and mathematical properties of the outcome of the algorithms. A boundary reconstruction of smooth convex sets through a second-order interpolation is possible with these properties. Furthermore, they allow different initial guess strategies to aim for fast convergence of trajectory recalculations.
The methods are not merely theoretical but can be applied to real-world scenarios. The mathematical model describing the behavior of a spacecraft during landing is formulated to prove this. For this scenario, the reachability set is calculated with the methods of this work and visualized. In the process, convexification is performed. The computation times and advantages of the presented algorithms are elaborated to consider the application of the results of this work in future space missions.

Keywords Reachability · Algorithms · Convex Optimization · Nonlinear Programming · Optimal Control · Parametric Sensitivity Analysis · Polytopes

Zusammenfassung

In dieser Arbeit werden optimierungsbasierte Algorithmen zur Approximation von Erreichbarkeitsmengen vorgestellt. Eine Erreichbarkeitsmenge umfasst alle Zustände eines dynamischen Systems, die durch zulässige Steuerungen bei einem gegebenen Anfangszustand erreicht werden. Ansätze, die auf der Optimierung und optimalen Steuerung basieren, haben dabei den Vorteil, dass neben der Dynamik und den Randbedingungen weitere Nebenbedingungen einbezogen werden können. Allerdings gilt das Lösen nichtlinearer Optimierungsaufgaben als zeitaufwendig, so dass dies gezielt geschehen sollte, um möglichst viele Erkenntnisse zu gewinnen.
Aus diesem Grund betten die vorgestellten Algorithmen die nötigen Optimierungsläufe in einen geometrischen Rahmen ein. Insbesondere werden in dieser Arbeit konvexe Mengen und Eigenschaften von Polytopen diskutiert, was eine strukturelle Vorstellung von geometrischen Objekten in höheren Dimensionen ermöglicht. Die Optimierungstheorie stellt das Fundament dieser Arbeit für die Behandlung nichtlinearer und konvexer Programme dar. Im dynamischen Kontext werden optimale Steuerprozesse eingeführt, die ein Kostenfunktional minimieren, das einer gewöhnlichen Differentialgleichung erster Ordnung und anderen Beschränkungen unterliegt. Die parametrische Sensitivitätsanalyse ist ein wesentlicher Bestandteil dieser Arbeit. Sie ist ein post-optimales Werkzeug, um ohne großen Aufwand zusätzliche Erkenntnisse aus einer gelösten Optimierung zu gewinnen.
Es werden drei Algorithmen zur Mengenapproximation vorgestellt. Sie basieren auf Gittern und Polytopen. Diese Arbeit deckt das Parallelisierungspotenzial der Algorithmen und die mathematischen Eigenschaften der Approximationsergebnisse auf. Mit diesen Eigenschaften ist eine Randrekonstruktion von glatten konvexen Mengen durch eine Interpolation zweiter Ordnung möglich. Darüber hinaus erlauben sie verschiedene Strategien für Startschätzungen, um eine schnelle Konvergenz bei Trajektorien-Neuberechnungen zu erreichen.
Die Methoden sind nicht nur theoretischer Natur, sondern können auch auf reale Szenarien angewendet werden. Um dies zu beweisen, wird ein mathematisches Modell formuliert, das das Verhalten eines Raumschiffs bei der Landung beschreibt. Für dieses Szenario wird die Erreichbarkeitsmenge mit den Methoden dieser Arbeit berechnet und visualisiert. Dabei wird eine Konvexifizierung durchgeführt. Es werden die Berechnungszeiten und die Vorteile der vorgestellten Algorithmen herausgearbeitet, um die Anwendung der Ergebnisse dieser Arbeit in zukünftigen Weltraummissionen zu erwägen.

Schlüsselwörter Erreichbarkeit · Algorithmen · Konvexe Optimierung · Nichtlineare Optimierung · Optimalsteuerung · Parametrische Sensitivitätsanalyse · Polytope

Danksagung

Die vorliegende Arbeit entstand im Rahmen des Networking Partnering Initiative der Europäischen Weltraumorganisation (ESA), durch die ich die einzigartige Chance erhalten habe, beim Deutschen Zentrum für Luft- und Raumfahrt (DLR) Bremen sowie an der Universität Bremen als wissenschaftlicher Mitarbeiter tätig zu sein. In der Abteilung für Navigations- und Regelungssysteme und in der Arbeitsgruppe für Optimierung und Optimale Steuerung (AG O2C) erfuhr ich stets eine offene und hilfsbereite Atmosphäre, durch die ich kritische und fördernde Rückmeldung für meine Arbeit erhalten habe. Ich bin sehr dankbar für diese tolle Zeit.

Ich möchte mich zunächst bei meinem Doktorvater Christof Büskens bedanken, der mit seiner Arbeit und Begeisterung für die Optimierung ein großartiges Vorbild für mich war, in unseren fachlichen Gesprächen immer die richtigen Fragen gestellt hat, um mich voranzubringen, und mir den Rücken stärkte, wann immer es nötig war. Auf DLR-Seite gilt mein Dank Stephan Theil und David Seelbinder, die mich ausgezeichnet betreut haben, mit mir die wildesten Ideen durchgegangen sind und meinen Blick für die Anwendung geschärft haben. Ganz besonders möchte ich an dieser Stelle Matthias Knauer herausheben, der mich sowohl fachlich als auch persönlich bestens unterstützt hat. Er hat mir aus verschiedensten Krisen herausgeholfen und war in der herausfordernden Zeit aufgrund der COVID-19-Pandemie stets für mich erreichbar. Danke!

Für mich ist die AG O2C in den letzten Jahren ein Arbeitsplatz geworden, an dem ich nicht nur produktiv und gegenseitig helfend arbeiten kann, sondern sie stellt auch einen Ort dar, auf den ich mich morgens freue. Dazu tragen zwar alle aus dem Kollegium bei, jedoch möchte ich einige wunderbare Persönlichkeiten gesondert nennen. Marcel Jacobse ist meine erste Anlaufstelle gewesen, wenn ich Hilfe beim Programmieren oder jemanden zum Diskutieren von Algorithmen und Details in der Optimierungstheorie brauchte. Maria Höffmann, Shruti Patel, Christian Meerpohl und Matthias Rick sind hilfsbereite und ebenfalls brillante KollegInnen, deren Meinung ich gerne einhole, und durch ihren Witz und ihre Intelligenz meinen Arbeitsalltag bereichern. Arne Berger bestärkte mich in meiner agilen Arbeitsweise und verdeutlichte mir, wie wertvoll meine Persönlichkeit und Expertise für ein Teamgefüge sein kann. Ich danke Euch allen für alles, was ich lernen und erleben durfte!

Ich möchte diese Arbeit meiner Familie widmen, die wenige Sekunden, nachdem ich ihnen mitgeteilt hatte, dass mir eine Promotion angeboten wurde, für mich entschieden hat, dass ich promovieren würde. Meinen Eltern Kwokliang und Fungnam, meiner Schwester Lina und meinem Bruder San Wah verdanke ich all die Liebe, positive Energie und Rückendeckung, die ich für dieses Unterfangen brauchte.

All das erhielt ich aber auch von Le Wang, meiner liebevollen Frau. Sie erlebte meine höchsten Höhen und tiefsten Tiefen der vergangenen Jahre und erinnerte mich stets an das Wesentliche im Leben. Egal, wie der Tag gelaufen ist, es wird ein guter Tag gewesen sein, wenn ich zu ihr nach Hause komme. Danke, dass es Dich gibt!

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

In recent years, space exploration has gained in attractiveness and importance pushed by private companies on the one hand. On the other hand, the questions of the origins of the universe, declining resources on Earth, and the imminent need for relocation due to climate change drive scientists worldwide to be engaged in space-related fields. Space projects are well funded and represent a chance for institutes or nations to enhance their prestige. Thus, hard- and software and procedures during mission application are designed in a highly but reasonably conservative manner attaching importance to reliability and robustness to ensure success. The standard for preparation is therefore high because once started, there are hardly any margins to deal with unforeseen circumstances.
Especially in deep-space exploration, when an uncrewed spacecraft is sent out, complete autonomy is pursued during mission design and realization. Controlling a spacecraft far from Earth is not viable because the signal propagation delay becomes too large. Consequently, decision-making must be conducted through a module of an on-board computer. This module must be supplied with data from equipped sensors and cameras processed through a relatively weak processor. Processors used in a spacecraft must withstand the solar radiation, and as a result of a trade-off, computational capacities are decreased.
The focus of this work is to design an efficient algorithm that holds the potential to compute so-called reachable sets in real-time. These sets contain states a dynamic system, i.e. a mathematical model of motion, can attain through feasible control inputs given its initial state. Provided this set, too risky maneuvers may be excluded and, instead, the most promising course of action may be chosen by the decision-making module. In rare cases, reachable sets can be explicitly stated, but mostly they are approximated with numerical and computative means. These are highly complex computations in which, theoretically, all possible control configurations over time must be propagated. In addition, strict requirements for reliability and real-time capability must be fulfilled.
The challenging hardware constraints do not prevail on the roads, and expensive methods to implement reachability analysis in autonomous driving are already state-of-the-art.

Typically, this application scenario considers fewer degrees of freedom, which aids in practicability. The reachability set is only used in the space domain's mission design and analysis phase. In the critical phase of a mission, pre-calculated boundaries are tested to determine whether or not a target can be reached under the prevailing conditions. These boundaries are based on strong simplifications that estimate a reachability set overly conservatively. If an algorithm provides the reachability set under real-time requirements, mission flexibility and robustness of a spacecraft in terms of guidance, navigation, and control can be significantly enhanced. A target may be evaluated concerning costs and risk and the influence of uncertainties on the current system state.
From a mathematical point of view, calculating a reachability set or a sufficiently accurate approximation is an appealing challenge. Mission requirements must be strictly adhered to in addition to a highly complex mathematical model. An approach using optimization and optimal control to address this task appears promising in this regard. With this powerful and versatile tool, those dynamic and mission constraints are respected, and it also optimizes in a definable aspect. The landing trajectory of a spaceship, for example, can be calculated with the help of optimization in the most resource-saving way. However, for reachability analysis, many such trajectories or optimization tasks would have to be determined or solved. Each optimization run is considered computationally expensive. For this reason, each optimization run should be as insightful as possible so that the number of expensive calculations can be kept low.

1.2 Reachability Analysis

The dynamic feasibility problem

$$\begin{aligned} &\dot{\boldsymbol{x}}(t) = f_{\text{dyn}}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \\ &C(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq 0_{n_c}, \quad \boldsymbol{x}(t_0) \in \mathcal{X}_0, \quad \boldsymbol{x}\left(t_f\right) \in \mathcal{X}_f, \\ &\boldsymbol{x}(t) \in \mathbb{R}^{n_x}, \quad \boldsymbol{u}(t) \in \mathbb{R}^{n_u}, \quad t \in \left[t_0, t_f\right] \end{aligned} \tag{1.1}$$

is considered in the reachability analysis in the scope of this work. It consists of a system dynamic f_{dyn} and path constraints C. A control $\boldsymbol{u}$ with n_u components is sought that leads to a trajectory $\boldsymbol{x}$ of n_x states. The process takes place in the interval $\left[t_0, t_f\right]$, the process time. Both trajectory and control must satisfy the constraints in the process time. For the state, there are boundary values through sets $\mathcal{X}_0$ and $\mathcal{X}_f$. The admissible final states form the so-called (forward) reachability set, which is defined as follows:

$$\begin{aligned} \mathcal{R}_{t_f}(x_0) := \{x_f \in \mathbb{R}^{n_x} \mid \exists \boldsymbol{u}(t) \in \mathbb{R}^{n_u} \wedge \exists \boldsymbol{x}(t) \in \mathbb{R}^{n_u} \ \forall t \in [0, t_f]: \\ \text{(1.1) holds with } \boldsymbol{x}(t_f) = x_f \text{ and } \mathcal{X}_0 = \{x_0\}\} \end{aligned} \tag{1.2}$$

In [6, 5], approaches to compute a reachable set are classified by the categories

i. set-valued methods,

ii. level set methods,

iii. optimization based algorithms,

and an overview is provided. The first two categories regards no path constraints and either $\mathcal{X}_0$ or $\mathcal{X}_f$ is a singleton, while the other is $\mathbb{R}^{n_x}$, thus does not represent a constraint. Set-valued methods are treated in [34, 47] among others, in which the focus lies in the reachable set of a time interval

$$\mathcal{R}_{[0,\tau]}(x_0) := \bigcup_{t\in[0,\tau]} \mathcal{R}_t(x_0) \tag{1.3}$$

subject to a linear dynamic f_{dyn}. If sets of initial states and controls are considered when solving a differential equation, one speaks of set-valued methods. Given an initial state $x_0 \in \mathbb{R}^{n_x}$ the solution for a linear time-invariant f_{dyn} can be analytically specified through a matrix exponential (cf. [68]). The initial values and controls are propagated over several steps with a time increment Δt. After each step, a reachability set is created and a union is eventually formed. If special geometries, such as zonotopes [4] or ellipsoids [46] are taken as a basis, set operations can be simplified. In case of nonlinear dynamics, linearization can be applied (cf. [3]). Another approach falling into the class of set-valued methods is based on the actual integration of all feasible controls which can be realized by parallelization on the GPU [57]. The underlying mathematical problem of set-valued methods can be formulated through differential inclusions in which the first-order ordinary differential equation in (1.1) is replaced by

$$\dot{\boldsymbol{x}}(t) \in \mathcal{F}(\boldsymbol{x}(t), t) \subset \mathbb{R}^{n_x}.$$

See [11] for more details. A short introduction to the subject of differential inclusions and their origins is given in [22].
In level set methods, an implicit surface function $\phi : \mathbb{R}^{n_x} \to \mathbb{R}$ is derived to specify the reachable set by

$$\phi(z) \begin{cases} \leq 0 & \text{for } z \in \mathcal{R}_{t_f}(x_0) \\ > 0 & \text{for } z \notin \mathcal{R}_{t_f}(x_0) \end{cases}.$$

In [52], backward reachability (or controllability) sets

$$\mathcal{C}_{t_0}(x_f) := \{x_0 \in \mathbb{R}^{n_x} \mid \exists \boldsymbol{u}(t) \in \mathbb{R}^{n_u} \wedge \exists \boldsymbol{x}(t) \in \mathbb{R}^{n_u} \ \forall t \in [0, t_f]: \\ (1.1) \text{ holds with } \boldsymbol{x}(t_0) = x_0 \text{ and } \mathcal{X}_f = \{x_f\}\} \tag{1.4}$$

are regarded, and it is shown that the implicit surface function ϕ characterizing the set is the viscosity solution of a Hamilton-Jacobi-Isaacs partial differential equation. Solutions of these PDEs are rather difficult to obtain, and this approach is considered only suitable for dynamic systems with up to four states (cf. [5]). Recently, as in [13], neural networks have been designed to solve these PDEs making the level set method more applicable for higher dimensional state spaces.
Optimization-based algorithms form the main topic of this work. In this class of methods, a cost functional is defined, which, together with (1.1) as the constraints, form an

optimal control problem (OCP). Unlike the other set-valued or level set methods, path constraints and constraints to the initial and final states are easily incorporated. Different procedures exist to approximate the forward or backward reachable sets depending on the cost functional. If the set is convex, an approach based on the idea of the support function is proposed by [10]. Elements on the $n_x - 1$-dimensional sphere are mapped to a support point that must lie on the boundary of the reachable set. This approach is extended in [26], where an over- and under-approximation given by polytopes is presented based on boundary elements of the set and supporting hyperplanes. In case of a non-convex reachable set, [12] suggests a grid approach. An equidistant grid is obtained by discretizing a subset of the state space. Each grid point is seen as a target a trajectory should lead to. Such a trajectory is found through a cost functional minimizing the distance between the final state and the grid point. If the optimal distance is (practically) zero, the corresponding grid point is an element of the reachable set. This approach is applied in [5], where the OCP is discretized using pseudo-spectral methods.

1.3 Outline of Work and Contribution

This work creates a sound foundation in geometry and optimization theory for the reader. This knowledge is necessary to examine existing optimization-based algorithms for the approximation of reachable sets and to explain their extensions. These algorithms are applied to simulative landing scenarios. Two models for lander dynamics are derived, which eventually lead to convex and nonlinear programs.
Chapter 2 introduces convex sets and their properties. They play a major role in the design of the algorithms in Chapter 4. Among others, simplices and convex polytopes are demonstrated as examples of convex sets. Concepts like the convex hull and terminology concerning polytopes are set up, simplifying explanations in later chapters. This chapter concludes with the definition of star-convexity and a star-shaped polytope.
At the beginning of Chapter 3, static optimization tasks are defined. An objective's minimizer is sought subject to equality and inequality constraints in these problems. The Lagrange methodology is used, and optimality conditions are derived. There are two classes of problems regarded in this context: nonlinear and convex programming. The SQP method and a primal-dual interior-point method are explained to solve mathematical programs. The solution of an optimization task can be further examined in a parametric sensitivity analysis. Parametric sensitivities are a practically free byproduct of an optimization that describes the solution's change subject to perturbations. Finally, optimal control problems are treated. They differ from static optimization problems in that the minimizer is sought in infinite-dimensional function spaces over a time interval. In this work, optimal control problems are first discretized and transformed into a nonlinear program. Single-step methods are demonstrated for discretization. Subsequently, the nonlinear program is solved using the methods presented in the first part of Chapter 3.
In Chapter 4, the knowledge of Chapters 2 and 3 is brought together, and algorithms are presented for the approximation of sets. At first, simple geometric bodies are defined to test the methods. Each approach is explained separately by describing the process, list-

ing the output properties, and discussing their advantages and disadvantages concerning their performances, reconstructing the examples. From the many necessary optimizations associated with each algorithm, more knowledge can be extracted than simply a solution of (1.1), which is the key message of Chapter 4.
Landing scenarios for spacecrafts are regarded in Chapter 5. In the first instance, a mathematical model is introduced and further processed in two different ways. In both cases, it is investigated where a spacecraft can land safely. In the first approach, a transformation is performed so that the three degrees of freedom that the final position state in space has, becomes two. The other approach is to relax the problem and other constraints so that convex programs are created that can be solved reliably and robustly using primal-dual interior-point methods.
The final Chapter 6 summarizes the critical aspects of this work and concludes. In the end, an outlook with open ends for further research is presented.

Acknowledgment

This research was conducted under No. 4000120111/NL/MH of the ESA Networking Partnering Initiative.

Chapter 2

Geometry of Polytopes in Higher Dimensions

In nature, all objects are perceived as three-dimensional. Through the senses of sight and touch, a humans can perceive these objects. They are able to process and convert them into two-dimensional entities like paintings or frames of a movie we project on screens. However, our conception has difficulties when we try to imagine higher dimensional objects. The same can be said about the sets this dissertation examines. A two- or a three-dimensional set is a planar, respectively spatial, object. We can depict one-dimensional sets with lines; a zero-dimensional set or a singleton can be exhibited through a point. When we increase the number of dimensions, the means to illustrate decreases, and we must employ mathematical constructs for orientation in higher dimensions. This chapter treats the necessary geometrical foundation to make a class of sets, so-called polytopes, more concrete, which challenge our conception due to their dimensionality.

2.1 Convex Sets

In general, sets with elements of the $\mathbb{R}^n$ can appear in arbitrary shapes. For example, they may be disconnected, have holes, or consist of discrete points. The focus of this section lies in *convex sets*. They are commonly introduced in the literature for convex optimization like [16] for instance, which deals with finding a minimizer of a function in a convex set. Convex sets also serve as a foundation for studying *polytopes* treated in the subsequent section.

Definition 2.1 (Affine and convex sets). *Let $y_1, y_2 \in \mathcal{C} \subseteq \mathbb{R}^n$ be two arbitrary points. $\mathcal{C}$ is*

i. affine if and only if

$$\alpha y_1 + (1-\alpha)\, y_2 \in \mathcal{C} \text{ for any } \alpha \in \mathbb{R} \text{ holds,}$$

ii. convex if and only if

$$\alpha y_1 + (1-\alpha)\, y_2 \in \mathcal{C} \text{ for any } \alpha \in [0,1] \text{ holds.}$$

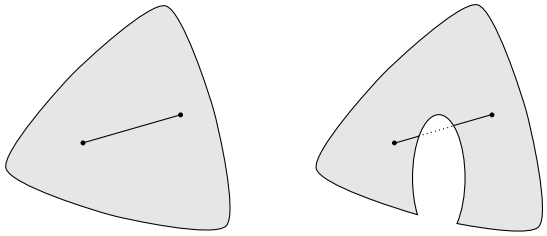


Figure 2.1: Convex (left) and non-convex (right) sets

In other words, for any two points of a convex set, the line segment connecting them is also contained in the set. Figure 2.1 illustrates this, where the left object is convex, and the right one is non-convex. If the set is affine, the line through both points is contained in the set. Thus, any affine set is also convex.

Example 2.2. *Consider the ball*

$$\mathcal{B}_\varepsilon(o) := \{y \in \mathbb{R}^n \mid \|y - o\| \leq \varepsilon\} \tag{2.1}$$

with center $o \in \mathbb{R}^n$ and radius $\varepsilon \geq 0$. Given $y_1, y_2 \in \mathcal{B}_\varepsilon(o)$ and $\alpha \in [0,1]$,

$$\|\alpha y_1 + (1-\alpha)\, y_2 - o\| \leq \alpha \|y_1 - o\| + (1-\alpha)\|y_2 - o\| \leq \varepsilon$$

holds due to the subadditivity and absolute homogeneity of the norm. Thus, we conclude $\alpha y_1 + (1-\alpha)\, y_2 \in \mathcal{B}_\varepsilon(o)$, and any ball is convex. The same applies to the open ball, i.e., the interior

$$\mathring{\mathcal{B}}_\varepsilon(o) := \{y \in \mathbb{R}^n \mid \|y - o\| < \varepsilon\}$$

but not for the sphere, i.e., the boundary

$$\partial\mathcal{B}_\varepsilon(o) := \{y \in \mathbb{R}^n \mid \|y - o\| = \varepsilon\}\,. \tag{2.2}$$

The points on the line segment connecting two elements of a convex set are so-called *convex combinations* of the two. Any finite number of the set's elements may form those combinations. An affine version exists as well.

Definition 2.3 (Affine and convex combination). *Given a finite number of points $y_1, \ldots, y_k \in \mathbb{R}^n$, $k \in \mathbb{N}$, the point*

$$\alpha_1 y_1 + \cdots + \alpha_k y_k \;\; \textit{with } \alpha_1 + \cdots + \alpha_k = 1$$

is called

i. an affine combination for $\alpha_1, \ldots, \alpha_k \in \mathbb{R}$.

ii. a convex combination for $\alpha_1, \ldots, \alpha_k \in [0,1]$.

The following lemma allows an alternate characterization of affine and convex sets through affine and convex combinations.

Lemma 2.4 (Characterization of affinity and convexity). *Some set $\mathcal{C} \subseteq \mathbb{R}^n$ is*

i. affine if and only if all affine combinations of elements in $\mathcal{C}$ are contained in $\mathcal{C}$,

ii. convex if and only if all convex combinations of elements in $\mathcal{C}$ are contained in $\mathcal{C}$.

Proof. A proof can be found in [7], p. 6. □

Lemma 2.4 generalizes Definition 2.1. According to this characterization, instead of only enclosing a line or a line segment, the corresponding set must contain more generic structures shaped by affine or convex combinations.
For some set $\mathcal{C} \subseteq \mathbb{R}^n$, its so-called *affine* or *convex hull* of $\mathcal{C}$ can be formed. In the literature, these terms are commonly introduced as the "smallest" affine or convex set that contains $\mathcal{C}$ (cf. [61]). The following is an equivalent definition.

Definition 2.5 (Affine and convex hull). *Given some set $\mathcal{C} \subseteq \mathbb{R}^n$,*

i. $\operatorname{aff}(\mathcal{C}) := \{\alpha_1 y_1 + \cdots + \alpha_k y_k \mid y_i \in \mathcal{C},\ \alpha_i \in \mathbb{R},\ i = 1, \ldots, k,\ \alpha_1 + \cdots + \alpha_k = 1,\ k \in \mathbb{N}\}$ *denotes the affine hull of $\mathcal{C}$.*

ii. $\operatorname{conv}(\mathcal{C}) := \{\alpha_1 y_1 + \cdots + \alpha_k y_k \mid y_i \in \mathcal{C},\ \alpha_i \in [0,1],\ i = 1, \ldots, k,\ \alpha_1 + \cdots + \alpha_k = 1,\ k \in \mathbb{N}\}$ *denotes the convex hull of $\mathcal{C}$.*

In other words, the convex (affine) hull of a set $\mathcal{C}$ consists of all convex (affine) combinations of elements in $\mathcal{C}$. If we look more closely at affine sets, we find that they are shifted linear subspaces (cf. Theorem 1.2 in [61]).

Proposition 2.6 (Affine sets are shifted linear subspaces). *Let $\mathcal{C}_{\text{aff}} \subseteq \mathbb{R}^n$ denote an affine set. For arbitrary $o \in \mathcal{C}_{\text{aff}}$, $\mathcal{C}_{\text{aff}} - o := \{y - o \mid y \in \mathcal{C}_{\text{aff}}\}$ is a linear subspace.*

Proof. By definition, linear subspaces are closed under addition and scalar multiplication. Choose $y_1, y_2 \in \mathcal{C}_{\text{aff}} - o$ and $\alpha, \beta \in \mathbb{R}$ arbitrarily. Note $y_1 + o \in \mathcal{C}_{\text{aff}}$ and $y_2 + o \in \mathcal{C}_{\text{aff}}$. Regard Lemma 2.4 and construct the affine combination

$$\begin{aligned} &\alpha(y_1 + o) + \beta(y_2 + o) + (1 - \alpha - \beta)\, o \in \mathcal{C}_{\text{aff}} \\ &\qquad\qquad \Leftrightarrow \alpha y_1 + \beta y_2 + o \in \mathcal{C}_{\text{aff}}. \end{aligned}$$

Consequently, $\alpha y_1 + \beta y_2 \in \mathcal{C}_{\text{aff}} - o$ holds, thus $\mathcal{C}_{\text{aff}} - o$ is a linear subspace. □

Linear spaces can be assigned a *dimension*. In the following, we would like to do the same with convex sets to establish the correct wording for the further course of the work.

Definition 2.7 (Dimension of convex sets). *Let $\mathcal{C} \subseteq \mathbb{R}^n$ be a convex set. Define $\mathcal{C}_{\text{aff}} := \operatorname{aff}(\mathcal{C})$. The dimension of $\mathcal{C}$ is defined as*

$$\dim \mathcal{C} := \dim(\mathcal{C}_{\text{aff}} - o) \quad \text{for any } o \in \mathcal{C}_{\text{aff}},$$

where $\mathcal{C}_{\text{aff}} - o := \{y - o \mid y \in C\}$. The dimension of the empty set is defined as -1.

The latter definition indicates why since the beginning, we have proposed the affine counterparts alongside, although this section's name suggests a focus on convex sets. Besides affine sets being shifted linear subspaces, we can observe that the following term establishes another analogy between vector and affine spaces.

Definition 2.8 (Affinely independent). *Points $y_0, \dots, y_k \in \mathbb{R}^n$, $k \in \mathbb{N}$, are affinely independent if and only if $y_1 - y_0, \dots, y_k - y_0$ are linearly independent.*

A k-dimensional vector space is generated by k linearly independent vectors, $k+1$ affinely independent vectors generate a k-dimensional affine space.

Proposition 2.9 (Preservation of convexity under affine transformation). *Let $\mathcal{C} \subseteq \mathbb{R}^n$ be a convex set and $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ specify an affine map $f_{\text{aff}} \colon \mathbb{R}^n \to \mathbb{R}^n$, $f_{\text{aff}}(y) = Ay + b$. Then, the image of $\mathcal{C}$ under f_{aff}*

$$f_{\text{aff}}(\mathcal{C}) := \{f_{\text{aff}}(x) \mid x \in \mathcal{C}\}$$

is convex.

Proof. For any two points in $f_{\text{aff}}(\mathcal{C})$, $y_1, y_2 \in C$ shall denote corresponding elements in their inverse image. For arbitrary $\alpha \in [0,1]$ it holds

$$\begin{aligned} &\alpha f_{\text{aff}}(y_1) + (1-\alpha) f_{\text{aff}}(y_2) = \alpha(Ay_1 + b) + (1-\alpha)(Ay_2 + b) \\ &= A(\alpha y_1 + (1-\alpha) y_2) + b = f_{\text{aff}}(\alpha y_1 + (1-\alpha) y_2) \end{aligned}$$

Due to convexity of C, C contains $\alpha y_1 + (1-\alpha) y_2$, thus $\alpha f_{\text{aff}}(y_1) + (1-\alpha) f_{\text{aff}}(y_2) \in f_{\text{aff}}(C)$ is valid. □

Hyperplanes are an example of affine spaces. They divide a vector space into two *halfspaces*.

Definition 2.10 (Hyperplane and halfspaces). *Let $a \in \mathbb{R}^n \setminus \{0\}$ and $\beta \in \mathbb{R}$. The set*

$$\mathcal{H} := \left\{y \in \mathbb{R}^n \mid a^\top y = \beta\right\}$$

is called hyperplane. Based on $\mathcal{H}$, two halfspaces can be defined:

$$\mathcal{H}_{\leq} := \left\{x \in \mathbb{R}^n \mid a^\top x \leq \beta\right\} \text{ and } \mathcal{H}_{>} := \left\{x \in \mathbb{R}^n \mid a^\top x > \beta\right\}.$$

$\mathcal{H}_{\leq}$ is called closed lower halfspace and $\mathcal{H}_{>}$ is called open upper halfspace. Analogously, $\mathcal{H}_{<}$ and $\mathcal{H}_{\geq}$ can be defined.

The true intention behind the introduction of hyperplanes is the *supporting hyperplane theorem*. A supporting hyperplane of a set touches the set's boundary such that the set is fully contained in one of the two halfspaces.

Definition 2.11 (Supporting hyperplane). *Given $\mathcal{C} \subset \mathbb{R}^n$, let y_0 be a point on the boundary $\partial\mathcal{C}$ of this set. If $a \in \mathbb{R}^n \setminus \{0\}$ exists such that $\mathcal{C} \subset \left\{y \in \mathbb{R}^n \mid a^\top y \leq a^\top y_0\right\}$, then $\left\{y \in \mathbb{R}^n \mid a^\top y = a^\top y_0\right\}$ is called a supporting hyperplane to $\mathcal{C}$ at y_0.*

For a convex set, a supporting hyperplane can be determined at every boundary point.

Theorem 2.12 (Supporting hyperplane theorem). *Let $\mathcal{C} \subset \mathbb{R}^n$ be a closed convex set with non-empty interior and y_0 be any point on the boundary of $\mathcal{C}$. There exists a supporting hyperplane to $\mathcal{C}$ at y_0.*

Proof. A proof can be found in [65], p.11. □

2.2 Polytopes

The previous section has presented the properties of convex sets. The convex hull as a mathematical operator has been introduced, which, provided some set $\mathcal{C}$, generates an enclosing convex set by supplementing it with all convex combinations viable with elements in $\mathcal{C}$. When $\mathcal{C}$ consists of a finite number of discrete points, its convex hull generates convex polytopes. Polytopes serve as means to construct two algorithms presented in Chapter 4. This section does not claim to cover the combinatorial nature within convex polytopes but illustrates them as geometrical objects. For a deeper treatment of the combinatorial aspects, it is referred to [72]. More geometric properties can be studied in [37, 65]. A classical reference book that provides an overview of the polytope theory is [38]. For further reads, [67] (especially chapters 15, 16, and 26) serves as a good starting point.

The convex hull of a discrete set of points is a polytope. The points that determine the shape of the polytope are called *vertices*.

Definition 2.13 (Vertex). *Given a set $\mathcal{C} \subset \mathbb{R}^n$ of discrete points, $v \in C$ is called a vertex if* $\operatorname{conv}(\mathcal{C} \setminus \{v\}) \neq \operatorname{conv}(\mathcal{C})$.

The corners of a cube are illustrative examples of vertices because the convex hull of the eight corners is the cube, but removing one results in a different object.

Definition 2.14 (Convex polytope). *Given a set of finite points $\mathcal{V} := \{v_0, \ldots, v_k\} \subset \mathbb{R}^n$, its convex hull $\mathcal{P} := \operatorname{conv}(\mathcal{V})$ is called convex polytope. If $\mathcal{V}$ is a set of vertices, $\mathcal{V}$ gives the $\mathcal{V}$-description of $\mathcal{P}$. If $\dim \mathcal{P} = n$, $\mathcal{P}$ is called full-dimensional.*

Certain parts of the boundary of a polytope result from the convex hulls of subsets of the set of vertices. Hence, they represent convex polytopes and are generally called *k-faces*.

Definition 2.15 (k-face). *Given a convex polytope $\mathcal{P} \subset \mathbb{R}^n$ and a hyperplane $\mathcal{H} \subset \mathbb{R}^n$ with $\mathring{\mathcal{P}} \cap \mathcal{H} = \emptyset$, $\partial\mathcal{P} \cap \mathcal{H} = \mathcal{F}$ and $\dim \mathcal{F} = k$, $\mathcal{F}$ denotes a k-face of $\mathcal{P}$. They are refered to as*

i. vertices in the 0-dimensional case.

ii. edges in the 1-dimensional case.

iii. ridges in the $(n-2)$-dimensional case.

iv. facets in the $(n-1)$-dimensional case.

Besides the $\mathcal{V}$-description of a convex polytope from Definition 2.14, there is also the *$\mathcal{H}$-description*. The latter allows the interpretation of a convex polytope as an intersection of halfspaces.

Proposition 2.16 ($\mathcal{H}$-description). *Let some vertex set $\mathcal{V} \subset \mathbb{R}^n$ denote the $\mathcal{V}$-description of the full-dimensional polytope $\mathcal{P} \subset \mathbb{R}^n$ that has $n_f \in \mathbb{N}$ facets. A matrix $A \in \mathbb{R}^{n_f \times n}$ and a vector $b \in \mathbb{R}^{n_f}$ exist such that*

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \leq b\}. \tag{2.3}$$

The right hand side of (2.3) *gives the $\mathcal{H}$-description of $\mathcal{P}$.*

Proof. A proof can be found in [72] in Section 1.1. □

In practice, matrix A and vector b of the $\mathcal{H}$-description of a full-dimensional convex polytope $\mathcal{P} \subset \mathbb{R}^n$ can be stated, given an interior point $o \in \mathring{\mathcal{P}}$ (cf. [70]). Assuming an arbitrary ordering of the n_f facets of $\mathcal{P}$, $\mathcal{F}_k \subset \partial\mathcal{P}$ shall denote the k-th facet. Select affinely independent points $y_1, \ldots, y_n \in \mathcal{F}_k$. Since $o \notin \text{aff}(\mathcal{F}_k)$, it holds that $\tilde{y}_i = y_i - o$ for $i = 1, \ldots, n$ are linearly independent due to Definition 2.8. The linear equation

$$Y_k a_k = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad \text{where } Y_k := \begin{bmatrix} \tilde{y}_1^\top \\ \vdots \\ \tilde{y}_n^\top \end{bmatrix}, \tag{2.4}$$

can be uniquely solved because Y_k is invertible. The transposed solution $a_k^\top$ is the k-th row of matrix A and $1 + a_k^\top o$ is the k-th component of the right hand side b yielding the $\mathcal{H}$-description. The question of how to choose affinely independent $y_1, \ldots, y_n$ of a facet does not arise as we will regard triangulated facets in the following, and the vertices of those triangles can be selected. This approach potentially leads to redundant inequalities of the $\mathcal{H}$-description, as a facet may be subdivided into multiple triangles. However, the polytope is wholly specified in this manner.
The $\mathcal{V}$- and the $\mathcal{H}$-description of a convex polytope are equivalent and can be transformed into the other. The process to receive the vertices of a polytope, described by inequalities, is called *vertex enumeration*. The task to find the $\mathcal{H}$-description, given the vertices of a polytope, is called *facet enumeration* or convex hull. Vertex and facet enumeration are dual problems the Avis-Fukuda algorithm solves in polynomial time and space, presented in [8]. Typically, triangular elements subdivide and specify the facets of a polytope. Let $n_\mathcal{V}$ denote the number of vertices and $n_\mathcal{F}$ the number of triangular subdivisions forming the facets of an n-dimensional convex polytope. Then, the following inequalities hold and are tight [8]:

$$n_\mathcal{V} \leq \phi(n, n_\mathcal{F}) \quad \text{and} \quad n_\mathcal{F} \leq \phi(n, n_\mathcal{V}) \tag{2.5}$$
$$\text{with } \phi(n,m) = \binom{m - \lceil \frac{n}{2} \rceil}{\lfloor \frac{n}{2} \rfloor} + \binom{m - 1 - \lceil \frac{n-1}{2} \rceil}{\lfloor \frac{n-1}{2} \rfloor}$$

The generalization of triangles in any dimension is called *simplex*. They represent an example of convex polytopes.

Definition 2.17 (k-simplex). *Let $k \leq n$ and $v_0, \ldots, v_k \in \mathbb{R}^n$ be affinely independent points. The polytope* $\text{conv}(\{v_0, \ldots, v_k\})$ *is called a k-simplex.*

Any convex hull of a subset of a simplex' set of vertices is a k-face in the form of a k-simplex. 0- to 3-dimensional simplices are illustrated in Figure 2.2. Simplices of higher dimensions are difficult to visualize but remain conceivable because each vertex is connected to another by an edge. Assume $v_0, \ldots, v_n \in \mathbb{R}^n$ denote the vertices of a full-dimensional simplex. The simplex' volume can be computed with the following formula:

$$\text{vol}(\text{conv}(\{v_0, \ldots, v_n\})) := \frac{1}{n!} \left| \det \begin{bmatrix} v_1 - v_0 & \ldots & v_n - v_0 \end{bmatrix} \right|. \tag{2.6}$$

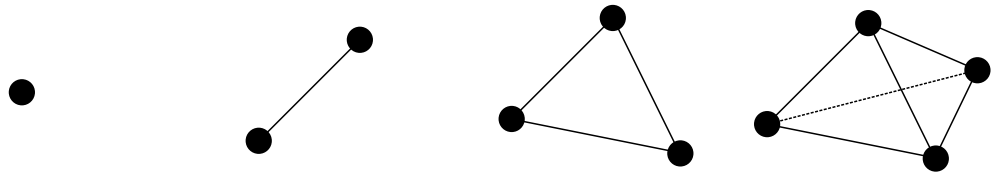


Figure 2.2: Examples of simplices of different dimensions

Simplices play an essential role in the following because they describe the boundary of a polytope. The notion of *ordered identifiers* is introduced to specify k-simplices in a set $\mathcal{V}$ of vertices. For this purpose, the elements in $\mathcal{V}$ are uniquely indexed so that $k+1$ numbers may be assigned to a k-simplex. These $k+1$ numbers are the indices of the vertices of the simplex.

Definition 2.18 (Ordered identifier). *Let* $\{v_1, \dots, v_{n_v}\} \subset \mathbb{R}^n$ *be a family of* $n_v \in \mathbb{N}$ *points and* $I \in \{1, \dots, n_v\}^{k+1}$ *denote a multi-index with* $k+1$ *ordered non-repetitive components* $I^j < I^{j+1}, j = 1, \dots, k$. *$I$ is called ordered identifier of k-simplex* $\mathcal{F} \subset \mathbb{R}^n$ *if and only if* $\mathcal{F} = \operatorname{conv}(v_{I^1}, \dots, v_{I^{k+1}})$.

Typically, a so-called facet-vertex incidence matrix describes the boundary of a polytope. However, the ordered identifier allows a cleaner presentation of the theory and can as well be used in the numerical realization of a polytope[1].
Besides convex polytopes, *star-shaped* sets are utilized and examined in the following chapters.

Definition 2.19 (Star-shaped set). *A set* $\mathcal{C} \subset \mathbb{R}^n$ *is called star-shaped if and only if* $o \in \mathcal{C}$ *exists such that, for any* $y \in \mathcal{C}$ *and* $\alpha \in [0,1]$, *it holds* $\alpha o + (1-\alpha)\, y \in \mathcal{C}$. *The point o is called vantage point.*

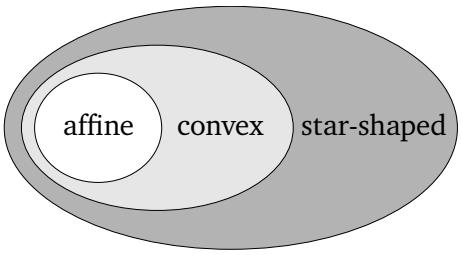


Figure 2.3: Relation of set properties

The term *star-convex* is a synonym for star-shaped. Similarities of Definitions 2.1 and 2.19 are obvious. Figure 2.3 illustrates the relation between the affine, convex, and star-shaped properties of a set. Any affine set is also convex since it contains all convex combinations of

[1]The ordered property is helpful to organize the multi-indices as keys in a hash map as duplicates can be easily detected.

its elements as it consists of all affine combinations. Any convex set is star-shaped, as any element can be the vantage point o.

Definition 2.20 (Star-shaped triangulated polytope). *Given a family of points $\mathcal{V} = \{v_1, \ldots, v_{n_v}\}$, let $\{I_i\}_{i=1,\ldots,n_f} \subset \{1, \ldots, n_v\}^n$ denote a non-repetitive sequence of ordered identifiers of $(n-1)$-simplices $\mathcal{F}_i = \operatorname{conv}\left(v_{I_i^1}, \ldots, v_{I_i^n}\right)$ with $\dim\left(\mathcal{F}_i \cap \mathcal{F}_j\right) < n-1$ for some $i, j = 1, \ldots, n_f$ with $i \neq j$.*
A set $\mathcal{P} \subset \mathbb{R}^n$ is called star-shaped triangulated polytope if and only if the following conditions are met:

i. $\mathcal{P}$ *is star-shaped,*

ii. $\partial\mathcal{P} = \bigcup_{i=1}^{n_f} \mathcal{F}_i$,

iii. $\mathring{\mathcal{P}} \neq \emptyset$.

The pair $\left(\mathcal{V}, \{I_i\}_{i=1,\ldots,n_f}\right)$ is called boundary description[2] of $\mathcal{P}$.

The type of polytopes described in Definition 2.20 is star-shaped and fully defined by a finite set of points and ordered identifiers that determine the simplicial subdivision of the boundary. Furthermore, it has a non-empty interior. The second condition assures that the boundary is completely assembled and has no holes. The terms introduced in Definition 2.15 will be used hereafter for any polytope that has a boundary description. Although strictly speaking they do not necessarily apply anymore because of the potential non-convexity, among other reasons, a conceptual structuring with k-faces is suitable.

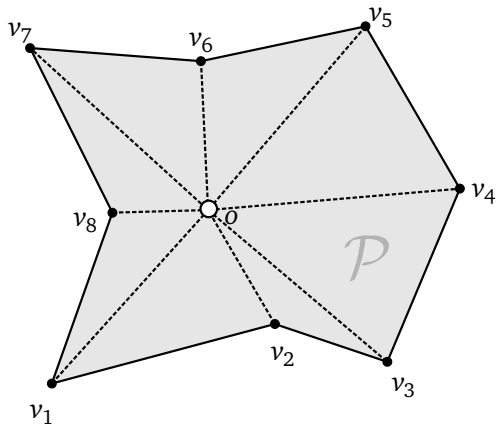


Figure 2.4: Example of a star-shaped triangulated polytope

[2]Although boundary description is introduced for more general polytopes, it is an established term for convex polytopes, also based on a triangulation (cf. [67], Chapter 26). Subdividing the boundary into simplices and identifying them is a common way to specify a polytope. For example, `Matlab`'s convex hull function works like this and returns identifiers to $(n-1)$-simplices of the triangulated boundary (cf. [14]).

Figure 2.4 illustrates a star-shaped triangulated polytope $\mathcal{P} \subset \mathbb{R}^2$. The points $v_1, \ldots, v_8$ represent vertices (and ridges since its a planar object). By $I_i = (i, i+1)$ for $i = 1, \ldots, 7$ and $I_8 = (1, 8)$ the sequence of ordered identifiers are given. They determine simplices $\mathcal{F}_i = \text{conv}\left(v_{I_i^1}, v_{I_i^2}\right)$ that form facets (or edges) of $\mathcal{P}$. By connecting the facets with a vantage point $o \in \mathcal{P}$, as illustrated in Figure 2.4, a star-shaped simplicial polytope is partitioned into full-dimensional simplices. This observation leads to the proposition about the volume of polytopes that concludes this chapter.

Proposition 2.21 (Volume of a star-shaped triangulated polytope). *Given the boundary description* $\left(\mathcal{V}, \{I_i\}_{i=1,\ldots,n_f}\right)$ *of a star-shaped triangulated polytope* $\mathcal{P} \subset \mathbb{R}^n$ *and a vantage point* $o \in \mathcal{P}$*, the volume of polytope is*

$$\text{vol}(\mathcal{P}) := \frac{1}{n!} \sum_{i=1}^{n_f} \left| \det \left[v_{I_i^1} - o \quad \ldots \quad v_{I_i^n} - o \right] \right|. \tag{2.7}$$

Proof. Since the intersections of full-dimensional simplex partitions of $\mathcal{P}$ have dimensions smaller than $n-1$, their volume equals 0. Thus, the volumes of the n-simplices add up to the volume of $\mathcal{P}$. The right hand side of (2.7) follows directly from (2.6). □

Chapter 3

Optimization and Optimal Control

This chapter covers the optimization theory, which will be necessary for the algorithms in Chapter 5. We distinguish between *static* and *dynamic* optimization. The first part is about solving nonlinear programs (NLP). These are minimization tasks with sufficiently smooth objective functions and constraints. SQP methods can solve them, approximating the problem quadratically in each iteration to approach a minimum successively. The sequentially obtained quadratic problems belong (under certain conditions) to the class of convex programs (CP) treated in a subsequent section. CPs, in turn, can be solved very efficiently with interior-point methods. We establish the fundamentals of parametric sensitivity analysis at the end.
The second part of this chapter deals with optimal control problems (OCP). Unlike in nonlinear programming, OCPs take a time-dependent process into account. A cost functional is minimized subject to a first-order ordinary differential equation (the dynamics), boundary values, and further general constraints. The direct method for solving OCPs is presented. A discretization leads to an NLP, which can then be solved using the approaches from the first part of this chapter.
Reference books for nonlinear programming are [33] and [54] among others, which have served as the main source. The latter also treats the post-optimal sensitivity analysis. [16] is the primary source for convex programming in the scope of this work. Static optimization is also covered in the context of optimal control problems in books such as [48] and [18].

3.1 Static Optimization

This section deals with constrained minimization problems of the following form

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{\mathrm{opt}}}} \quad & f(x) \\ \text{subject to} \quad & g(x) = 0, \\ & h(x) \leq 0, \end{aligned} \tag{OP}$$

where $f : \mathbb{R}^{n_{\text{opt}}} \to \mathbb{R}$ is called the *objective function* while $g : \mathbb{R}^{n_{\text{opt}}} \to \mathbb{R}^{n_g}$ and $h : \mathbb{R}^{n_{\text{opt}}} \to \mathbb{R}^{n_h}$ denote *equality* and *inequality constraints*. The function argument $x \in \mathbb{R}^{n_{\text{opt}}}$ is a vector that contains the *optimization variables*. Depending on the functions' properties, we can make different statements about (OP). In the following, general terms are established, and the method of *Lagrange multipliers* is presented. The minimization problem (OP) is separately regarded as convex and nonlinear programs. The behavior of a solution to a minimization problem under perturbations is the subject of the parametric sensitivity analysis, which concludes this section.

3.1.1 General Definitions

Unlike in unconstrained optimization, in which the minimizer of an objective function is sought in the whole $\mathbb{R}^{n_{\text{opt}}}$, (OP) involves constraints. Subsequently, an element is called *feasible* and is part of the *feasible set* if it fulfills those constraints.

Definition 3.1 (Feasibility). *Given the constrained minimization problem* (OP)*, a point* $x \in \mathbb{R}^{n_{opt}}$ *is called feasible if and only if*

$$g(x) = 0 \text{ and } h(x) \leq 0$$

are fulfilled. $\mathcal{X} := \{x \in \mathbb{R}^{n_{opt}} \mid g(x) = 0 \text{ and } h(x) \leq 0\}$ *is referred to as the feasible set.*

An element in the feasible set that solves (OP) is generally called a *minimizer*. When we study algorithms that solves optimization tasks, we distinguish between *global* and *local* minimizer.

Definition 3.2 (Global and local minimizer). *Given the constrained minimization problem* (OP)*, a feasible point* $x^* \in \mathbb{R}^{n_{opt}}$ *is called*

i. *a global minimizer if and only if for all feasible* $x \in \mathbb{R}^{n_{opt}} \setminus \{x^*\}$
ii. *a local minimizer if a neighborhood* $\mathcal{N}(x^*) \subset \mathbb{R}^{n_{opt}}$ *exists such that for all feasible* $x \in \mathcal{N}(x^*) \setminus \{x^*\}$

the inequality

$$f(x^*) \leq f(x) \tag{3.1}$$

holds. A feasible point x^* *is a strict minimizer if and only if the strict inequality is fulfilled in* (3.1).

The inequalities of (OP) are regarded differently compared to the equality constraints. An inequality may be *active* or not.

Definition 3.3 (Active constraints and active set). *Given a feasible point* $x \in \mathbb{R}^{n_{opt}}$ *of the constrained minimization problem* (OP)*, let* $i \in \{1, \ldots, n_h\}$ *denote some index. The i-th inequality constraint is called active if and only if* $h_i(x) = 0$*. The set* $\mathcal{A}(x) := \{i \in \{1, \ldots, n_h\} : h_i(x) = 0\}$ *contains all active inequality constraints and is referred to as the active set.*

In constrained optimization, the *Lagrange function* is defined as the sum of the objective and weighted constraints.

Definition 3.4 (Lagrange function). *Given the constrained minimization problem* (OP)*, the* Lagrange function $L : \mathbb{R}^{n_{opt}} \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h} \to \mathbb{R}$ *is defined as*

$$L(x, \lambda, \mu) := f(x) + \lambda^\top g(x) + \mu^\top h(x) \tag{3.2}$$

where $\lambda \in \mathbb{R}^{n_g}, \mu \in \mathbb{R}^{n_h}$ *are called Lagrange multipliers.*

The Lagrange multipliers are also called *dual variables*, while x represents the primal variables. Accordingly, a dual problem to (OP) exists.

Definition 3.5 (Dual problem). *Given the constrained minimization problem* (OP) *and its Lagrange function* $L : \mathbb{R}^{n_{opt}} \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h} \to \mathbb{R}$ *as in* (3.2)*, let* $q : \mathbb{R}^{n_g} \times \mathbb{R}^{n_h} \to \mathbb{R}$ *be a function defined as*

$$q(\lambda, \mu) := \inf_{x \in \mathbb{R}^{n_{opt}}} L(x, \lambda, \mu).$$

Then, the dual problem has the following form:

$$\begin{aligned} \max_{\lambda \in \mathbb{R}^{n_g}, \mu \in \mathbb{R}^{n_h}} \quad & q(\lambda, \mu) \\ \textit{subject to} \quad & \mu \geq 0 \end{aligned} \tag{DP}$$

For (OP) and its dual problem, the following relationship holds.

Theorem 3.6 (Weak duality). *Let* $x^* \in \mathbb{R}^{n_{opt}}$ *and* $\lambda^* \in \mathbb{R}^{n_g}$*,* $\mu^* \in \mathbb{R}^{n_h}$ *be the optimal solutions of the primal problem* (OP) *and dual problem* (DP) *respectively. Then, the following inequality holds:*

$$q(\lambda^*, \mu^*) \leq f(x^*). \tag{3.3}$$

This property is called weak duality.

Proof. For arbitrary λ and $\mu \geq 0$ and a feasible $\bar{x}$, $q(\lambda, \mu) \leq L(\bar{x}, \lambda, \mu)$ holds. Due to feasibility, $\lambda^\top g(\bar{x}) + \mu^\top h(\bar{x}) \leq 0$ holds. Thus, $L(\bar{x}, \lambda, \mu) \leq f(\bar{x})$ follows. This is especially true for $\lambda = \lambda^*, \mu = \mu^*$ and $\bar{x} = x^*$, and therefore $q(\lambda^*, \mu^*) \leq f(x^*)$ applies (cf. [16], p.216). □

There also exists the term *strong duality* in this context.

Definition 3.7 (Strong Duality). *Let* $x^* \in \mathbb{R}^{n_{opt}}$ *and* $\lambda^* \in \mathbb{R}^{n_g}$*,* $\mu^* \in \mathbb{R}^{n_h}$ *be the optimal solutions of the primal problem* (OP) *and dual problem* (DP) *respectively. Strong duality is said to hold for* (OP) *if and only if*

$$q(\lambda^*, \mu^*) = f(x^*) \tag{3.4}$$

applies.

If strong duality holds for (OP), a characterization for optimality can be formulated based on a triple (x, λ, μ) and the definition of the so-called *duality gap*

$$d^*(x, \lambda, \mu) := f(x) - q(\lambda, \mu),$$

which must equal 0 for $(x, \lambda, \mu) = (x^*, \lambda^*, \mu^*)$. Primal-dual interior-point methods, which are presented in Section 3.1.3.1 for convex programs, takes the duality gap into account.

3.1.2 Nonlinear Programming

In this section, the functions in (OP) are assumed to be sufficiently smooth. Under these conditions, (OP) is referred to as a *nonlinear program*.

Definition 3.8 (Nonlinear program (NLP)). *Let $f: \mathbb{R}^{n_{opt}} \to \mathbb{R}$, $g: \mathbb{R}^{n_{opt}} \to \mathbb{R}^{n_g}$ and $h: \mathbb{R}^{n_{opt}} \to \mathbb{R}^{n_h}$ be twice continuously differentiable. The minimization task*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & f(x) \\ \textit{subject to} \quad & g(x) = 0, \\ & h(x) \leq 0 \end{aligned} \tag{NLP}$$

is called nonlinear program.

Unlike in global optimization, solving (NLP) is equivalent to finding a local minimizer in the scope of this work. The so-called *Karush-Kuhn-Tucker* conditions play a central role in the following and are formulated based on the differentiability of f, g, and h.

Definition 3.9 (Karush-Kuhn-Tucker (KKT) conditions). *Let $L: \mathbb{R}^{n_{opt}} \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h} \to \mathbb{R}$ denote the Lagrange function of (NLP). Moreover, let $x \in \mathcal{X}$, $\lambda \in \mathbb{R}^{n_g}$ and $\mu \in \mathbb{R}^{n_h}_{\geq 0}$. Then*

$$\nabla_x L(x, \lambda, \mu) = \nabla_x f(x) + \lambda^\top \nabla_x g(x) + \mu^\top \nabla_x h(x) = 0 \tag{3.5a}$$
$$\nabla_\lambda L(x, \lambda, \mu) = g(x) = 0 \tag{3.5b}$$
$$\nabla_\mu L(x, \lambda, \mu) = h(x) \leq 0 \tag{3.5c}$$
$$L(x, \lambda, \mu) - f(x) = h(x)^\top \mu = 0 \tag{3.5d}$$

are called Karush-Kuhn-Tucker (KKT) conditions. A triple (x^*, λ^*, μ^*), for which (3.5) holds, is called a KKT point.

The KKT conditions were published in 1951 and were first known as Kuhn-Tucker conditions [44]. The results were already described by William Karush in 1939 in his master's thesis [40], which was not published.
In the following, we will learn under which premise the KKT conditions are the first-order necessary optimality conditions. A local minimizer must comply with these conditions. For an unconstrained problem, the KKT conditions imply the necessary optimality conditions for a local minimizer: The requirement that the gradient of the cost function is 0 is provided by (3.5a), while the other terms which concern the constraints, as well as (3.5b), (3.5c), and (3.5d) are omitted. In constrained optimization, feasibility is claimed by (3.5b) and (3.5c). The last condition (3.5d) is referred to as *complementarity*. A point that fulfills the KKT conditions is termed *critical* but is not necessarily a minimizer. In addition, the constraint must possess certain properties known as *constraint qualifications* (CQ). Sufficient optimality conditions exist as well and will also be presented. After that, an SQP algorithm is studied, which uses the smoothness of NLPs to find a critical point.

3.1.2.1 Optimality Conditions

The following explanations are based on the reference book [33]. Page references lead to all proofs necessary for this subsection. According to Definition 3.2, a solution x^* of (NLP) is considered locally optimal if the objective function f does not decrease in its feasible neighborhood. In the unconstrained case, the gradient of the objective disappears while the Hessian is positive semidefinite in x^*, i.e.

$$\nabla f(x^*) = 0 \quad \text{and} \quad \nabla^2 f(x^*) \geq 0.$$

In the constrained case, it might not be possible to find an equilibrium for ∇f. Instead, finding a solution means to determine a point $x^* \in \mathcal{X}$ where any infinitesimal step to another feasible element leads to either growth or no change in the objective. The *tangent cone* of $\mathcal{X}$ in x^* contains all feasible directions for these steps.

Definition 3.10 (Tangent cone). *Let $\mathcal{C} \subseteq \mathbb{R}^n$ denote a non-empty set. Then, for an element $y \in \mathcal{C}$,*

$$\mathcal{T}_{\mathcal{C}}(y) = \left\{ d \in \mathbb{R}^n \mid \exists \left\{ y^k \right\} \subseteq \mathcal{C},\ \exists \{t_k\} \subset \mathbb{R} \colon\ y^k \to y \text{ and } \left(y^k - y\right)/t_k \to d \text{ for } t_k \searrow 0 \right\}$$

is called the tangent cone of $\mathcal{C}$ in y.

The tangent cone is closed (cf. [33], page 42), which, together with the mean value theorem, proves the following first version of the necessary first-order optimality condition.

Lemma 3.11. *Given* (NLP) *with a non-empty feasible set $\mathcal{X}$, let x^* denote a local minimizer of (NLP). Then,*

$$\nabla f(x^*)^\top d \geq 0$$

applies for all $d \in \mathcal{T}_{\mathcal{X}}(x^*)$.

Proof. See [33], p. 43. □

Lemma 3.11 states that the objective increases in all feasible directions from a local minimizer x^*. Unfortunately, $\mathcal{T}_{\mathcal{X}}(x^*)$ is rather inconvenient to work with in practice, whereas the *linearized tangent cone* is easier to grasp.

Definition 3.12 (Linearized tangent cone). *Given a feasible point $x \in \mathcal{X}$ and the active set $\mathcal{A}(x)$, the set*

$$\mathcal{T}_{lin}(x) := \left\{ d \in \mathbb{R}^{n_{opt}} \colon\ \nabla g_j(x)^\top d = 0, j \in \left\{1, \ldots, n_g\right\}, \nabla h_i^\top d \leq 0, i \in \mathcal{A}(x) \right\} \tag{3.6}$$

is called linearized tangent cone of $\mathcal{X}$ in x.

For a feasible point $x \in \mathcal{X}$ for which the tangent cone $\mathcal{T}_{\mathcal{X}}(x)$ and its linearized variant $\mathcal{T}_{lin}(x)$ coincide, the following term is introduced.

Definition 3.13 (Abadie constraint qualification). *A feasible point $x \in \mathcal{X}$ is said to comply with the Abadie constraint qualification if and only if $\mathcal{T}_{\mathcal{X}}(x) = \mathcal{T}_{lin}(x)$.*

With the Abadie CQ and Lemma (3.11), the following first-order necessary optimality condition is stated, which incorporates the KKT conditions from Definition 3.9.

Theorem 3.14 (KKT conditions with Abadie CQ). *Let $x^* \in \mathbb{R}^{n_{opt}}$ denote a local minimizer of* (NLP) *which complies with the Abadie CQ. Then, Lagrange multipliers $\lambda^* \in \mathbb{R}^{n_g}$ and $\mu^* \in \mathbb{R}^{n_h}$ exist such that (x^*, λ^*, μ^*) is a KKT point.*

Proof. See [33], p. 48. □

There exist requirements that are easier to check than the Abadie CQ but imply optimality like Theorem 3.14. The *linear independence constraint qualification* is probably the best known and commonly mentioned condition in this context.

Definition 3.15 (Linear independence constraint qualification (LICQ)). *Let $x \in \mathcal{X}$ be a feasible point of (NLP) and $\mathcal{A}(x)$ be the active set. The point x is said to comply with the linear independence constraint qualification if and only if the gradient of the equality and active inequality constraints*

$$\nabla g_i(x),\ i = 1, \ldots, n_g, \ \text{ and } \ \nabla h_i(x),\ i \in \mathcal{A}(x),$$

are linear independent.

Unlike in Theorem 3.14, the uniqueness of the Lagrange multipliers for a given minimizer is ensured through LICQ.

Theorem 3.16 (KKT conditions with LICQ). *Let $x^* \in \mathbb{R}^{n_{opt}}$ denote a local minimizer of (NLP) which complies with the LICQ. Then unique Lagrange multipliers $\lambda^* \in \mathbb{R}^{n_g}$ and $\mu^* \in \mathbb{R}^{n_h}$ exist such that (x^*, λ^*, μ^*) is a KKT point.*

Proof. See [33], p. 53. □

The *critical cone* is defined to formulate second-order necessary and sufficient optimality conditions.

Definition 3.17 (Critical cone). *Given a KKT point (x^*, λ^*, μ^*) of (NLP), the critical cone is defined as*

$$\mathcal{T}_{crit}(x^*, \lambda^*, \mu^*) = \big\{ d \in \mathcal{T}_{lin}(x^*) : \nabla g_i(x^*)^\top d = 0, i = 1, \ldots, n_g, \\ \text{and } \nabla h_i(x^*)^\top d = 0, i = 1, \ldots, n_h, \ \text{with } \mu_i^* > 0 \big\}.$$

The critical cone is a subset of the linearized tangent cone. It consists of search directions along which the active set remains unchanged. The following two theorems use the critical cone and the second-order derivatives of the objective and constraints in Definition 3.8.

Theorem 3.18 (Second-order necessary optimality conditions). *Given a local minimizer $x^* \in \mathcal{X}$ of (NLP) for which LICQ holds, let $\lambda^* \in \mathbb{R}^{n_g}, \mu^* \in \mathbb{R}^{n_h}$ be the Lagrange multipliers that lead to a KKT point (x^*, λ^*, μ^*). Let $L : \mathbb{R}^{n_{opt}} \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h} \to \mathbb{R}$ denote the Lagrange function of (NLP). Then,*

$$d^\top \nabla_x^2 L(x^*, \lambda^*, \mu^*)\, d \geq 0$$

holds for all $d \in \mathcal{T}_{crit}(x^*, \lambda^*, \mu^*)$.

Proof. See [33], pp. 65-66. □

As the last theorem concluding this subsection, the second-order sufficient optimality conditions are stated.

Theorem 3.19 (Second-order sufficient optimality conditions). *Let (x^*, λ^*, μ^*) be a KKT point of (NLP) for which*

$$d^\top \nabla_x^2 L(x^*, \lambda^*, \mu^*)\, d > 0$$

holds for all $d \in \mathcal{T}_{crit}(x^*, \lambda^*, \mu^*), d \neq 0$. Then x^* is a strict local minimizer of (NLP).

Proof. See [33], pages 67-68. □

3.1.2.2 Sequential Quadratic Programming

The previous subsection has shown the relationship between a KKT point and the optimality conditions. This section is about identifying critical points. The method presented in this work is called the *sequential quadratic programming* (SQP) method. The underlying idea is to approximate (NLP) quadratically in each iteration and, as the name suggests, to solve quadratic programs repeatedly. The solution of the quadratic problem of an iteration gives a direction to approach the sought (local) minimizer. A quadratic program is thereby defined as follows.

Definition 3.20 (Quadratic program). *Let $Q \in \mathbb{R}^{n_{opt} \times n_{opt}}$ denote a symmetric matrix. Moreover, let $A_g \in \mathbb{R}^{n_g \times n_{opt}}$, $b_g \in \mathbb{R}^{n_g}$, $A_h \in \mathbb{R}^{n_h \times n_{opt}}$ and $b_h \in \mathbb{R}^{n_h}$. Then,*

$$\min_{x \in \mathbb{R}^n} \quad \frac{1}{2} x^\top Q x + c^\top x \tag{3.7a}$$

$$\text{subject to} \quad A_g x - b_g = 0 \tag{3.7b}$$

$$A_h x - b_h \leq 0 \tag{3.7c}$$

is called a quadratic program.

We define an equality constrained optimization problem, which we will use to understand the basic idea of the SQP method:

$$\min_{x \in \mathbb{R}^n} \quad f(x) \tag{3.8a}$$
$$\text{subject to} \quad g(x) = 0 \tag{3.8b}$$

For (3.8), a critical point is sought, for which the Lagrange function is necessary:

$$L(x, \lambda) = f(x) + \lambda^\top g(x) \tag{3.9}$$

The KKT conditions for (3.8) are satisfied if

$$0 = \begin{pmatrix} \nabla_x L(x, \lambda) \\ g(x) \end{pmatrix} =: F(x, \lambda) \tag{3.10}$$

holds. In other words, a root of the function $F : \mathbb{R}^{n_{\text{opt}}} \times \mathbb{R}^{n_g} \to \mathbb{R}^{n_{\text{opt}}+n_g}$ must be determined. For this, the Newton method is used, in which a solution is approached iteratively. It yields the following iteration rule for function F given an initial guess $\left(x^0, \lambda^0\right) \in \mathbb{R}^{n_{\text{opt}}} \times \mathbb{R}^{n_g}$:

$$\nabla_{(x,\lambda)} F\left(x^k, \lambda^k\right) \begin{pmatrix} x^{k+1} - x^k \\ \lambda^{k+1} - \lambda^k \end{pmatrix} = -F\left(x^k, \lambda^k\right), \tag{3.11}$$

which is equivalent to

$$\nabla_{(x,\lambda)} F\left(x^k, \lambda^k\right) \begin{pmatrix} x^{k+1} - x^k \\ \lambda^{k+1} \end{pmatrix} = -F\left(x^k, \lambda^k\right) + \nabla_{(x,\lambda)} F\left(x^k, \lambda^k\right) \begin{pmatrix} 0 \\ \lambda^k \end{pmatrix}. \tag{3.12}$$

Regarding (3.9) and (3.10), the Jacobian can be specified as

$$\nabla_{(x,\lambda)} F\left(x^k, \lambda^k\right) = \begin{bmatrix} \nabla_x^2 L(x, \lambda) & \nabla_x g(x)^\top \\ \nabla_x g(x) & 0 \end{bmatrix}. \tag{3.13}$$

By inserting (3.10) and (3.13) into (3.12),

$$\begin{bmatrix} \nabla_x^2 L(x, \lambda) & \nabla_x g(x)^\top \\ \nabla_x g(x) & 0 \end{bmatrix} \begin{pmatrix} x^{k+1} - x^k \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla_x f\left(x^k\right) \\ -g\left(x^k\right) \end{pmatrix} \tag{3.14}$$

is obtained. Introducing substitution variable $\Delta x := x^{k+1} - x^k$, (3.14) represents the KKT conditions of the equality-constrained quadratic program

$$\min_{\Delta x \in \mathbb{R}^{n_{\text{opt}}}} \quad \frac{1}{2} \Delta x^\top \nabla_x^2 L\left(x^k, \lambda^k\right) \Delta x + \nabla_x f\left(x^k\right)^\top \Delta x \tag{3.15a}$$
$$\text{subject to} \quad g\left(x^k\right) + \nabla_x g\left(x^k\right)^\top d = 0 \tag{3.15b}$$

The equality constraints in (3.15b) are the first-order Taylor approximations of the actual equality constraints in (3.8) at the point x^k. In [42], the question is answered of how to include the inequalities in NLP, but we skip the details at this point. Following one's

intuition, the inequality constraints are also linearized and incorporated in the following quadratic program.

$$\min_{\Delta x \in \mathbb{R}^{n_{\text{opt}}}} \quad \frac{1}{2}\Delta x^\top \nabla_x^2 L\left(x^k, \lambda^k, \mu^k\right)\Delta x + \nabla_x f\left(x^k\right)^\top \Delta x \tag{3.16a}$$

$$\text{subject to} \quad g\left(x^k\right) + \nabla_x g\left(x^k\right)^\top \Delta x = 0 \tag{3.16b}$$

$$h\left(x^k\right) + \nabla_x h\left(x^k\right)^\top \Delta x \leq 0 \tag{3.16c}$$

Task (3.16) is solved in every iteration of the local SQP method which is summarized in Algorithm A.

Algorithm A Local SQP Method

A-1: *(Initialization)* Choose $\left(x^0, \lambda^0, \mu^0\right) \in \mathbb{R}^{n_{\text{opt}}} \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h}$ and set $k := 0$
A-2: *(Optimality?)* If $\left(x^k, \lambda^k, \mu^k\right)$ is a KKT point of (NLP): STOP.
A-3: *(Compute step)* Solve (3.16) to get $\left(\Delta x, \lambda^{k+1}, \mu^{k+1}\right)$
A-4: *(Update)* Set $x^{k+1} := x^k + \Delta x$ and increment $k := k+1$, go to step A-2.

If there is more than one KKT point for (3.16) in A-3, the one that is *closest* to the previous iterate is chosen. In practice, it is rather difficult to determine the actual closest KKT point. However, suppose we could do that, then, the following Theorem can be stated.

Theorem 3.21 (Convergence of local SQP method). *Let (x^*, λ^*, μ^*) denote a KKT point of (NLP) for which*

i. strict complementarity,

ii. LICQ,

iii. second-order sufficient optimality condition

hold. Then, a neighborhood $\mathcal{N}(x^*, \lambda^*, \mu^*)$ exists such that for any $\left(x^0, \lambda^0, \mu^0\right) \in \mathcal{N}(x^*, \lambda^*, \mu^*)$ the following holds:

i. The sequence $\left\{\left(x^k, \lambda^k, \mu^k\right)\right\}$ generated by Algorithm A converges to (x^*, λ^*, μ^*).

ii. The convergence rate is superlinear.

iii. If $\nabla_x^2 f$, $\nabla_x^2 g_i, i = 1, \ldots, n_g$ and $\nabla_x^2 h_i, i = 1, \ldots, n_h$ are locally Lipschitz-continuous, the convergence rate is quadratic.

Proof. See [33], p. 246. □

A step size control can be included in A-4 of Algorithm A that scales the step to the next iterate: $x^{k+1} = x^k + \sigma \Delta x$. The task of finding a suitable $\sigma > 0$ is also known as *line search*, and it aims to sufficiently reduce the objective function in every iterate. A backtracking step size control will be presented in Section 3.1.3.1 in the context of a primal-dual interior-point method that solves convex problems. A detailed treatise of line search methods, including a convergence study, is given in Chapter 3 of [54].

3.1.3 Convex Programming

In contrast to NLP, convex programs have stricter requirements for the objective and constraints. This section will treat (OP), in which f and the components of h are *convex*, while g is *affine*.

Definition 3.22 (Convex, concave, and affine functions). *The scalar function $f : \mathbb{R}^n \to \mathbb{R}$ is called*

i. convex if and only if for all $x, y \in \mathbb{R}^n$ and for all $\alpha \in [0,1]$ it holds:

$$f(\alpha x + (1-\alpha) y) \leq \alpha f(x) + (1-\alpha) f(y)$$

ii. strictly convex if and only if for all $x, y \in \mathbb{R}^n$, $x \neq y$ and for all $\alpha \in (0,1)$ it holds:

$$f(\alpha x + (1-\alpha) y) < \alpha f(x) + (1-\alpha) f(y)$$

iii. (strictly) concave if and only if $-f$ is (strictly) convex

A vector valued function $g : \mathbb{R}^n \to \mathbb{R}^m$ is said to be affine if and only if $A_g \in \mathbb{R}^{m \times n}$ and $b_g \in \mathbb{R}^n$ exist such that for all $x \in \mathbb{R}^n$ it holds

$$g(x) = A_g x + b_g.$$

Based on these properties, we can define a *convex program*.

Definition 3.23 (Convex program). *Let $f : \mathbb{R}^{n_{opt}} \to \mathbb{R}$ and $h : \mathbb{R}^{n_{opt}} \to \mathbb{R}^{n_h}$ be convex and $A_g \in \mathbb{R}^{n_g \times n_{opt}}$, $b_g \in \mathbb{R}^{n_g}$. The minimization task*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & f(x) \\ \text{subject to} \quad & A_g x + b_g = 0, \\ & h(x) \leq 0 \end{aligned} \tag{CP}$$

is called convex program.

The following two optimization tasks belong to the class of convex programs.

Example 3.24 (Linear and convex quadratic program). *Linear and convex quadratic programs are commonly known optimization tasks which are convex programs.*

i. Given a linear objective function $f : \mathbb{R}^{n_{opt}} \to \mathbb{R}^{n_h}$, $x \mapsto c^\top x$ and $h : \mathbb{R}^{n_{opt}} \to \mathbb{R}^{n_h}$, $x \mapsto A_h x + b_h$ representing affine inequality constraints, (CP) *is a linear program:*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & c^\top x \\ \text{subject to} \quad & A_g x + b_g = 0, \\ & A_h x + b_h \leq 0, \end{aligned} \tag{LP}$$

with $c \in \mathbb{R}^{n_{opt}}$, $A_g \in \mathbb{R}^{n_g \times n_{opt}}$, $b_g \in \mathbb{R}^{n_g}$, $A_h \in \mathbb{R}^{n_h \times n_{opt}}$, and $b_h \in \mathbb{R}^{n_h}$.

ii. Given a convex and quadratic objective function $f : \mathbb{R}^{n_{opt}} \to \mathbb{R}$, $x \mapsto x^\top Q x + c^\top x$ *and* $h : \mathbb{R}^{n_{opt}} \to \mathbb{R}^{n_h}$, $x \mapsto A_h x + b_h$ *representing affine inequality constraints,* (CP) *is a convex quadratic program:*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & x^\top Q x + c^\top x \\ \text{subject to} \quad & A_g x + b_g = 0, \\ & A_h x + b_h \leq 0, \end{aligned} \tag{CQP}$$

where $Q \in \mathbb{R}^{n_{opt} \times n_{opt}}$ *is symmetric and positive semi-definite,* $c \in \mathbb{R}^{n_{opt}}$, $A_g \in \mathbb{R}^{n_g \times n_{opt}}$, $b_g \in \mathbb{R}^{n_g}$, $A_h \in \mathbb{R}^{n_h \times n_{opt}}$, *and* $b_h \in \mathbb{R}^{n_h}$.

The quadratic program (3.16) introduced for the SQP algorithm has the same form as (CQP). The latter requires the matrix Q to be positive semi-definite. This property simplifies the optimality check, as the Hessian of the Lagrange function of (CQP) is always positive semi-definite.
The convexity of (CP) is connected to the convexity in a geometric sense, as stated in the following lemma.

Lemma 3.25 (Convex feasible set). *The feasible set* $\mathcal{X}$ *of* (CP) *is convex.*

Proof. Let $x, y \in \mathcal{X}$ denote feasible points. Any element $\alpha x + (1-\alpha) y$ for $\alpha \in [0,1]$ is feasible because

$$g(\alpha x + (1-\alpha) y) = A_g(\alpha x + (1-\alpha) y) + b_g = \alpha\left(A_g x + b_g\right) + (1-\alpha)\left(A_g y + b_g\right) = 0$$

due to affinity and

$$h(\alpha x + (1-\alpha) y) \leq \alpha h(x) + (1-\alpha) h(y) \leq 0$$

due to convexity hold (see Definition 3.22). Thus, the feasible set $\mathcal{X}$ of (CP) is convex according to Definition 2.1. □

The convexity and affinity of the functions in a CP are strong conditions. Analogously compelling statements can be derived from these assumptions. For example, given a local minimizer of a CP, it follows that this minimizer is global.

Proposition 3.26 (Global minimizer of CP). *Let* $x^* \in \mathbb{R}^{n_{opt}}$ *be a local minimizer of a* (CP). *Then,* x^* *is a global minimizer.*

Proof. Suppose $\tilde{x} \in \mathcal{X}$ exists for which

$$f(\tilde{x}) < f(x^*)$$

holds. The point $\alpha\tilde{x} + (1-\alpha) x^*$ for any $\alpha \in [0,1]$ is feasible as shown in Lemma 3.25. Thus, since f is convex, it follows

$$\begin{aligned} f(\alpha\tilde{x} + (1-\alpha) x^*) &\leq \alpha f(\tilde{x}) + (1-\alpha) f(x^*) \\ &< \alpha f(x^*) + (1-\alpha) f(x^*) = f(x^*). \end{aligned}$$

For $\alpha \to 0$ the point $\alpha\tilde{x} + (1-\alpha) x^*$ must be an element of the neighborhood $\mathcal{N}(x^*)$ of x^* that exists according to Definition 3.2. This contradicts the fact that x^* is a local minimizer. □

3.1.3.1 Interior-Point Methods

Interior-point methods were studied and developed in the 1950s, and 1960s [31]. In their application to linear programs, however, they were always overshadowed by the simplex method developed by George B. Dantzig in 1947 [69]. In 1984, Karmarkar's work [39] triggered a new enthusiasm for interior point methods because it presented a method with polynomial complexity for solving linear programs. Another milestone was the work [53] by Nesterov and Nemirovskii in 1994, who were able to prove the polynomial complexity result for more general convex programs. A more detailed historical overview of interior-points methods and what impact the developments had to different problem classes can be found in [69, 60, 36].
In this section, the primal-dual interior-point method of [16] is presented that solves convex programs (CP) with sufficiently smooth convex objective f and inequalities h. Due to the differentiability of these functions, the KKT-conditions in Definition 3.9 can be exploited again. In this method, the root of the residual function $r_\kappa \colon \mathbb{R}^{n_{\text{opt}}} \times \mathbb{R}^{n_h} \times \mathbb{R}^{n_g} \to \mathbb{R}^{n_{\text{opt}}+n_h+n_g}$ with components

$$r_\kappa(x,\mu,\lambda) = \begin{pmatrix} r_{\text{dual}}(x,\mu,\lambda) \\ r_{\text{cent}}(x,\mu,\lambda) \\ r_{\text{primal}}(x,\mu,\lambda) \end{pmatrix} \tag{3.17}$$

defined by

$$r_{\text{dual}}(x,\mu,\lambda) := \nabla_x f(x) + \lambda^\top A_g + \mu^\top \nabla_x h(x), \tag{3.18a}$$
$$r_{\text{cent}}(x,\mu,\lambda) := -\operatorname{diag}(\mu) h(x) - \kappa^{-1}\mathbf{1}, \tag{3.18b}$$
$$r_{\text{primal}}(x,\lambda,\mu) := A_g x + b_g. \tag{3.18c}$$

is repeatedly approached with updated $\kappa > 0$. If simultaneously $h_i(x) \le 0$ and $\mu_i \ge 0$ for all $i = 1,\dots,n_h$ can be ensured besides $r_\kappa(x,\lambda,\mu) = 0$, the KKT conditions (3.5) in a modified form are fulfilled. The modification happens in the complementarity (3.18b). In comparison to (3.5d), the product $-h_i(x)\mu_i$ is not 0, but κ^{-1} which implies that the inequalities are not active, and the corresponding Lagrange multipliers are positive. The roots of (3.17) with varying κ are therefore points in the interior of the primal and dual feasible set, which gives this method its name. With growing κ, it is possible to get arbitrarily close to the original KKT-conditions and the boundary of the primal and dual feasible set. In order to approach the root of the residual function r_κ, Newton's method yields a direction to follow. The actual root is not necessarily computed but a step towards it. As in (3.10) and (3.11),

$$\nabla_{(x,\mu,\lambda)} r_\kappa(x,\mu,\lambda) \begin{pmatrix} \Delta x \\ \Delta\mu \\ \Delta\lambda \end{pmatrix} = -r_\kappa(x,\mu,\lambda) \tag{3.19}$$

determines the direction of the step. Equation (3.19) is equivalent to

$$\begin{bmatrix} \nabla_x^2 f(x) + \sum_{i=1}^{n_h} \nabla_x^2 h_i(x) & \nabla h(x) & A_g^\top \\ -\operatorname{diag}(\mu)\nabla h(x) & -\operatorname{diag}(h(x)) & 0 \\ A_g & 0 & 0 \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta\mu \\ \Delta\lambda \end{pmatrix} = -r_\kappa(x,\mu,\lambda). \tag{3.20}$$

Provided a step size $\sigma > 0$, the solution of this linear equation system leads to the new iterate

$$(x, \mu, \lambda) + \sigma (\Delta x, \Delta\mu, \Delta\lambda), \tag{3.21}$$

which is checked whether it fulfills some termination criteria. The step size σ is determined by a line search. A *backtracking step size control* is described in Algorithm B.

Algorithm B Backtracking Step Size Control

B-1: *(Input)*
  B-1.1: *(Current iterate and search direction)*
    – (x, μ, λ) with $h_i(x) < 0$ and $\mu_i > 0$, $i = 1, \ldots, n_h$
    – $(\Delta x, \Delta\mu, \Delta\lambda)$ from (3.20)
  B-1.2: *(Parameters)* $\alpha > 0$ and $\beta \in (0, 1)$

B-2: *(Largest step size)* Determine

$$\bar{\sigma} := \sup \{\sigma \in [0, 1] : \mu_i + \sigma \Delta\mu_i \geq 0,\ i = 1, \ldots, n_h\}$$

and set $\sigma := 0.99\bar{\sigma}$

B-3: *(Loop)* Until

$$h_i(x + \Delta x) < 0,\ i = 1, \ldots, n_h, \text{ and } \|r_\kappa(x + \sigma\Delta x, \mu + \sigma\Delta\mu, \lambda + \sigma\Delta\lambda)\| \leq (1 - \alpha\sigma)\|r_\kappa(x, \mu, \lambda)\|$$

repeat $\sigma := \beta\sigma$

Provided the input, in the first instance, the largest step size $\bar{\sigma}$ is determined such that the Lagrange multipliers corresponding to the inequalities remain non-negative. An initial $\sigma = 0.99\bar{\sigma}$ is chosen that leads to strictly positive components of $\mu + \sigma\Delta\mu$. In B-3, σ is repeatedly multiplied by an a priori chosen parameter $\beta \in (0, 1)$ until the inequalities are strictly fulfilled, and a certain progress towards the root of the residual function (3.17) is ensured with the next step. The progress is expressed with another parameter $\alpha > 0$. Since the primal and dual variables are elements of finite-dimensional space, any norm could be utilized for the backtracking step size control, as all norms are equivalent under these circumstances. In case of the Euclidean norm $\|\cdot\|_2$, typically in practice, α and β are chosen in the range from 0.01 to 0.1 and 0.3 to 0.8, respectively (cf. [16]).

Algorithm C describes a primal-dual interior-point method and summarizes the specifications of this section. The algorithm expects primal and dual variables as inputs, which strictly fulfills the inequalities of the primal and dual problems. Actual primal feasibility is unnecessary but will be reached in the further progression. Constants $\varepsilon_{\text{feas}}$ and $\varepsilon_{\text{opti}}$ are selected to prescribe termination criteria. Both parameters are typically small. Furthermore, due to the parameter ν, κ successively grows to near the original KKT conditions. As long as the norm of the residuals r_{primal} and r_{dual} and the complementarity do not fall below $\varepsilon_{\text{feas}}$ and $\varepsilon_{\text{opti}}$, κ is updated, and a step towards the root of the adjusted residual function r_κ is performed.

Algorithm C is based on ideas of Newton's method, as in many applications. The biggest hurdle lies in the inequality constraints, which are overcome by a suitable step size control like Algorithm B in the presented method. In other interior-point methods, *barrier functions* are introduced, which go to infinity if their argument becomes non-negative. If the

argument is negative, it maps to values that are nearly 0. Each inequality of the optimization problem is concatenated with a barrier function and added to the objective function as a weighted penalty term. Typically, a differentiable barrier $\phi : \mathbb{R}_+ \to \mathbb{R}$ based on the logarithm function is chosen with $\phi(y) = -\log(-y)$. The optimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{\text{opt}}}} \quad & f(x) + \kappa^{-1} \sum_{i=1}^{n_h} \phi(h_i(x)) \\ \text{subject to} \quad & A_g x + b_g = 0 \end{aligned} \tag{3.22}$$

is convex, and for $h(x) < 0$ sufficiently smooth to apply Newton's method to determine KKT points. An approach in which problem (3.22) is sequentially solved with growing κ is called a *primal barrier method*.
In interior-point methods for linear and convex quadratic programs, a partitioning of the search direction (as in C-2.2) into a *predictor* and *corrector* step has proven its worth in practice. The predictor step ensures progress towards a (modified) KKT point while the corrector step addresses the error due to the linearization as in (3.19). Mehrotra introduced this idea in [51] for linear programs and Nocedal and Wright extended it for convex quadratic programs (compare [54], pp. 479).

Algorithm C Primal-Dual Interior-Point Method

C-1: *(Input)*
- C-1.1: *(Primal and dual variables)* $x \in \mathbb{R}^{n_{\text{opt}}}$ with $h_i(x) < 0$, $\mu_i > 0$ for $i = 1, \ldots, n_h$ and $\lambda \in \mathbb{R}^{n_g}$
- C-1.2: *(Parameters)* $\varepsilon_{\text{feas}}, \varepsilon_{\text{opti}} > 0$ and $\nu > 1$

C-2: *(Loop)* Until $\left\| r_{\text{primal}}(x, \mu, \lambda) \right\| \leq \varepsilon_{\text{feas}}$, $\| r_{\text{dual}}(x, \mu, \lambda) \| \leq \varepsilon_{\text{feas}}$ and $\vartheta := -h(x)^\top \mu \leq \varepsilon_{\text{opti}}$, repeat
- C-2.1: *(Modification factor)* Set $\kappa := \nu n_h / \vartheta$
- C-2.2: *(Search direction)* Compute primal-dual search direction $(\Delta x, \Delta \mu, \Delta \lambda)$ according to (3.20)
- C-2.3: *(Step size control and update)*
  - Determine step size $\sigma > 0$ with Algorithm B
  - Set $(x, \mu, \lambda) := (x, \mu, \lambda) + \sigma(\Delta x, \Delta \mu, \Delta \lambda)$

3.1.3.2 Convex Programs with Generalized Inequalities

For a long time, mathematicians considered linear and nonlinear programs utterly separate optimization fields and developed individual solution strategies and analyses for them. With the progress of inner-point methods in the 1980s, rethinking happened, and linear and nonlinear convex programs were more and more united. The cone-based inequalities introduced in works such as [49] in the 1960s were revisited in the course of this.
This section shows the formulation of *conic programs*, states their optimality conditions and defines the so-called *second-order cone programs*. For this purpose, we first define *proper cones*.

Definition 3.27 (Proper cone). *A set $\mathcal{K} \subseteq \mathbb{R}^n$ is called cone if and only if for any $y \in \mathcal{K}$, $\alpha y \in \mathcal{K}$ holds for all $\alpha \geq 0$. Furthermore, a cone $\mathcal{K}$ is called proper if the following conditions are fulfilled:*

i. $\mathcal{K}$ *is convex,*

ii. $\mathcal{K}$ *is closed,*

iii. $\mathring{\mathcal{K}} \neq \emptyset$*,*

iv. $y \in \mathcal{K}$ *and* $-y \in \mathcal{K}$ *imply* $y = 0$*.*

A proper cone induces a partial ordering used in *generalized inequalities*.

Definition 3.28 (Generalized inequality). *Let $\mathcal{K} \subseteq \mathbb{R}^n$ denote a proper cone and $y, z \in \mathbb{R}^n$. The expressions*

$$y \preceq_{\mathcal{K}} z \text{ and } z \succeq_{\mathcal{K}} y$$

are equivalent to $z - y \in \mathcal{K}$ and called generalized inequality. The strict generalized inequality is given by

$$y \prec_{\mathcal{K}} z \text{ and } z \succ_{\mathcal{K}} y$$

and is equivalent to $z - y \in \mathring{\mathcal{K}}$.

Provided these definitions, we formulate convex programs with generalized inequalities.

Definition 3.29 (Convex program with generalized inequalities). *Let $f : \mathbb{R}^{n_{opt}} \to \mathbb{R}$ and $h_i : \mathbb{R}^{n_{opt}} \to \mathbb{R}^{n_i}$, $i = 1, \ldots, n_K$ be convex and $A_g \in \mathbb{R}^{n_g \times n_{opt}}$, $b_g \in \mathbb{R}^{n_g}$. Furthermore, let $\mathcal{K}_i \subset \mathbb{R}^{n_i}$ denote proper cones for all $i = 1, \ldots, n_K$. The minimization task*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & f(x) \\ \text{subject to} \quad & A_g x + b_g = 0 \\ & h_i(x) \preceq_{\mathcal{K}_i} 0, \quad i \in \{1, \ldots, n_K\} \end{aligned} \tag{CPG}$$

is called convex program with generalized inequalities.

The obvious difference to CP is that the partial ordering induced by proper cones replaces the "$\leq$". The optimization task (CPG) is also treated with Lagrange multipliers. The dual variables corresponding to the generalized inequalities are sought in the respective *dual cones*.

Definition 3.30 (Dual and self-dual cone). *Let $\mathcal{K} \subseteq \mathbb{R}^n$ be a cone. The set*

$$\mathcal{K}^* = \left\{ y \in \mathbb{R}^n : z^\top y \geq 0 \text{ for all } z \in \mathcal{K} \right\}$$

is called dual cone of $\mathcal{K}$. $\mathcal{K}$ is called self-dual if and only if $\mathcal{K} = \mathcal{K}^*$.

The *positive orthant* and *second-order cones* are examples of self-dual cones. They are of particular interest because they define the generalized inequalities of second-order cone programs.

Example 3.31 (Positive orthant and second-order cone). *The two sets*

i. $\mathbb{R}_+ := \{y \in \mathbb{R} : y \geq 0\}$*, positive orthant,*

ii. $\mathcal{Q}^{n+1} := \{(y_0, y_1) \in \mathbb{R} \times \mathbb{R}^n : y_0 \geq \|y_1\|_2\}$*, the second-order cone,*

are self-dual cones[1].

Note that (CP) can be regarded as (CPG), in which all $\mathcal{K}_i$ are positive orthants. Therefore, the following constraint qualification and optimality conditions also apply to general convex programs.

Definition 3.32 (Slater constraint qualification). *Consider a convex program with generalized inequalities* (CPG). (CPG) *is said to fulfill the Slater constraint qualification if and only if* $x \in \mathbb{R}^{n_{opt}}$ *exists such that* $h_i(x) \prec_{\mathcal{K}_i} 0$ *and* $A_g x + b_g = 0$.

The Slater constraint qualification is fulfilled if the interior of the feasible set of (CPG) is not empty. Furthermore, if f and h_i, $i = 1, \ldots, n_K$ are sufficiently smooth, the KKT conditions are necessary and sufficient for optimality.

Theorem 3.33 (KKT conditions with Slater constraint qualification). *Assume that the Slater's constraint qualification holds for* (CPG) *with differentiable functions* f *and* h_i, $i = 1, \ldots, n_K$. *The triple* (x^*, λ^*, μ^*) *minimizes the convex program with generalized inequalities if and only if the generalized KKT conditions*

$$\begin{aligned} \nabla_x f(x^*) + A_g^\top \lambda^* + \sum_{i=1}^{n_K} \nabla_x h_i(x^*)^\top \mu_i^* &= 0, \\ A_g x^* + b_g &= 0, \\ h_i(x^*) &\preceq_{\mathcal{K}_i} 0, \quad i = 1, \ldots, n_K \\ \mu_i^* &\succeq_{\mathcal{K}_i^*} 0, \quad i = 1, \ldots, n_K \\ h_i(x^*)^\top \mu_i^* &= 0, \quad i = 1, \ldots, n_K \end{aligned} \tag{3.23}$$

hold.

Proof. See [16], pp. 264. □

The second-order cone program is an optimization task with a linear objective function and generalized inequalities induced by either positive orthants or second-order cones.

[1] In the literature, the set of symmetric positive definite matrices $\mathcal{S}_{++}^n$ is often introduced as an example for self-dual cones as well. $\mathcal{S}_{++}^n$ induces a partial ordering used in semidefinite programs, which represent a field that is closely related to linear or second-order cone programming following the unifying theory based on Euclidean Jordan algebra [2, 29].

Definition 3.34 (Second-order cone program). *Let K be the direct product of n_K positive orthants and second-order cones. Furthermore, let $c \in \mathbb{R}^{n_{opt}}$, $A_g \in \mathbb{R}^{n_g \times n_{opt}}$, $b_g \in \mathbb{R}^{n_g}$, $A_h \in \mathbb{R}^{n_h \times n_{opt}}$ and $b_h \in \mathbb{R}^{n_h}$ be given. The minimization task*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & c^\top x \\ \text{subject to} \quad & A_g x = b_g \\ & A_h x \preceq_{\mathcal{K}} b_h \end{aligned} \tag{SOCP}$$

is called second-order cone program

The class of second-order cone programs contains linear programs. However, it covers even more problem types. As an example, *quadratically constrained quadratic programs* can also be transformed into an SOCP.

Example 3.35 (Quadratically constrained quadratic programs). *Let $Q_0, \dots, Q_{n_h} \in \mathbb{R}^{n_{opt} \times n_{opt}}$ be symmetric and positive semidefinite, $a_1, \dots, a_{n_h} \in \mathbb{R}^{n_{opt}}$ and $\beta_1, \dots, \beta_{n_h} \in \mathbb{R}$. The quadratically constrained quadratic program*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & x^\top Q_0 x + c^\top x \\ \text{subject to} \quad & A_g x + b_g = 0, \\ & x^\top Q_i x + a_i^\top x + \beta_i \le 0, \quad i \in \{1, \dots, n_h\}, \end{aligned} \tag{QCQP}$$

is a second-order cone program. Provided the square root $\tilde{A}_i \in \mathbb{R}^{n_{opt} \times n_{opt}}$ of Q_i, i.e. $Q_i = \tilde{A}_i^\top \tilde{A}_i$, $i = 1, \dots, n_h$, this can be shown with the equivalent formulation of the quadratic inequality

$$x^\top Q_i x + a_i^\top x + \beta_i \le 0 \quad \Leftrightarrow \quad \left\| \begin{matrix} \frac{1}{2}\left(1 + a_i^\top x + \beta_i\right) \\ \tilde{A}_i x \end{matrix} \right\|_2 \le \frac{1}{2}\left(1 - a_i^\top x - \beta_i\right)$$

as the first step. The i-th inequality in (QCQP) *can be generalized in the following way*

$$A_i x \preceq_{\mathcal{K}_i} b_i, \quad \text{with } A_i := \begin{bmatrix} \frac{1}{2} a_i^\top \\ -\frac{1}{2} a_i^\top \\ -\tilde{A}_i \end{bmatrix} \text{ and } b_i := \begin{pmatrix} \frac{1}{2} - \frac{1}{2}\beta_i \\ \frac{1}{2} + \frac{1}{2}\beta_i \\ 0 \end{pmatrix}, \tag{3.24}$$

where $\mathcal{K}_i$ is a second-order cone. In order to "linearize" the objective function of (QCQP) *introduce a new optimization variable τ and the constraint*

$$x^\top Q_0 x + c^\top x \le \tau. \tag{3.25}$$

Doing the same procedure shown in (3.24) *for* (3.25) *leads to a corresponding A_0 and b_0 concluding the transformation to an SOCP.*

There are interior-point methods that solve SOCPs with an iteration complexity of $\sqrt{n}$ for problems with n second-order cone constraints [2]. Besides the global solution of convex programs addressed in Proposition 3.26, primal and dual infeasibility and thus, the

solvability can be certificated for an SOCP based on its self-dual reformulation [71]. Furthermore, an expert initial guess in order to solve an SOCP is not necessary [24]. The successful application of convex optimization in the aerospace domain, including planetary landing, rendezvous, and docking scenarios, was advanced in [9] among others. An extensive overview regarding the applications and convexification approaches is given in [50].

3.1.4 Parametric Sensitivities

In the last two sections, we studied nonlinear and convex programs presented as specializations of (OP). Optimality conditions were studied, and methods were presented that can be applied to both classes of problems, respectively. In this section, we learn about parametric sensitivity analysis. Once the minimizer $x^* \in \mathbb{R}^{n_{\text{opt}}}$ of (OP) is found, the parametric sensitivity analysis is concerned with understanding how the solution behaves under perturbations. For this purpose, another quantity is added in the formulation of an optimization task, which can be regarded as a design parameter or directly as a perturbation.

Definition 3.36 (Perturbed optimization program). *Suppose that $x \in \mathbb{R}^{n_{opt}}$, $p \in \mathbb{R}^{n_p}$, $f : \mathbb{R}^{n_{opt}} \times \mathbb{R}^{n_p} \to \mathbb{R}$, $g : \mathbb{R}^{n_{opt}} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_g}$ and $h : \mathbb{R}^{n_{opt}} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_h}$. Then,*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & f(x,p) \\ \text{subject to} \quad & g(x,p) = 0, \\ & h(x,p) \leq 0 \end{aligned} \tag{3.26}$$

is referred to as a perturbed optimization program. Given a reference value $p = p_0 \in \mathbb{R}^{n_p}$, (3.26) *is called nominal problem.*

If one were able to explicitly specify a function $x : \mathbb{R}^{n_p} \to \mathbb{R}^{n_{\text{opt}}}$ such that $x(p)$ is the solution of (3.26) for a perturbation $p \in \mathbb{R}^{n_p}$, valuable information would be available. However, to actually evaluate $x(\cdot)$ at $p \in \mathbb{R}^{n_p}$, one usually has no choice but to solve the perturbed optimization program by numerical means. However, if $x^* \in \mathbb{R}^{n_{\text{opt}}}$ represents the minimizer of the nominal problem of (3.26) for $p = p_0 \in \mathbb{R}^{n_p}$, in addition to $x(p_0) = x^*$, insights about $x(\cdot)$ and its derivative $\frac{\mathrm{d}}{\mathrm{d}p}x$ for a neighborhood of p_0 can be obtained.

Theorem 3.37 (Implicit functions). *Let $F : \mathbb{R}^n \times \mathbb{R}^{n_p} \to \mathbb{R}^n$, $(y,p) \mapsto F(y,p)$ denote a continuously differentiable function. Assume for a pair $(y^*, p_0) \in \mathbb{R}^n \times \mathbb{R}^{n_p}$ that $F(y^*, p_0) = 0$ applies. Moreover, assume that the Jacobian $\frac{\mathrm{d}}{\mathrm{d}y}F(y^*, p_0) \in \mathbb{R}^{n \times n}$ is invertible. Then, open neighborhoods $\mathcal{N}_y(y^*) \subset \mathbb{R}^n$ and $\mathcal{N}_p(p_0) \subset \mathbb{R}^{n_p}$ of y^* and p_0 as well as a continuosly differentiable function $y : \mathcal{N}_p(p_0) \to \mathbb{R}^n$ exist such that*

i. $y(p_0) = y^*$,

ii. $F(y(p), p) = 0$ *for all* $p \in \mathcal{N}_p(p_0)$.

If $F(y,p) = 0$ holds for a pair $(y,p) \in \mathcal{N}_y(y^*) \times \mathcal{N}_p(p_0)$, $y = y(p)$ follows. Furthermore, the following holds:

$$\frac{\mathrm{d}}{\mathrm{d}p}y(p_0) = \left(\frac{\mathrm{d}}{\mathrm{d}y}F(y^*, p_0)\right)^{-1} \frac{\mathrm{d}}{\mathrm{d}p}F(y^*, p_0) \in \mathbb{R}^{n \times n_p}. \tag{3.27}$$

Proof. A proof can be found on pages 95-98 in [32]. □

Theorem 3.37 is central in the proof of the subsequent sensitivity theorem. Suppose a nominal solution (x^*, λ^*, μ^*) of (3.26) for $p = p_0$ is given that fulfills the second-order necessary conditions of Theorem 3.18. Apply Theorem 3.37 by setting $y = (x, \lambda, \mu)$ and defining

$$F(x, \lambda, \mu, p) := \begin{pmatrix} \nabla_x L(x, \lambda, \mu, p) \\ g(x, p) \\ \operatorname{diag}(\lambda_1, \ldots, \lambda_{n_h}) h(x, p) \end{pmatrix} = 0 \tag{3.28}$$

in (3.27), where $L : \mathbb{R}^{n_{\text{opt}}} \times \mathbb{R}^{n_g} \times \mathbb{R}^{n_h} \times \mathbb{R}^{n_p} \to \mathbb{R}$ denotes the Lagrange function of the perturbed problem (3.26). The expression (3.28) reflects the KKT conditions from Definition 3.9 with additional dependency on perturbation p. In this context, Theorem 3.37 states how the nominal solution of a sufficiently smooth perturbed optimization program changes as a function of the perturbation. These derivatives are called *sensitivity derivatives* or just *sensitivities*. A linear system of equations has to be solved to obtain them. If stronger conditions are assumed, the following theorem gives even more insights into the implicit functions of the primary and dual variables of (3.26).

Theorem 3.38 (Sensitivity theorem). *Let (x^*, λ^*, μ^*) fulfill the second-order sufficient optimality conditions for a nominal solution of* (3.26) *with $p = p_0$. For active inequality constraints $h_i(x^*, p_0), i \in \mathcal{A}(x^*)$, $\mu > 0$ is assumed. Moreover, for the objective and the constraint functions of* (3.26)*, suppose that*

i. f, g, h are twice continuously differentiable subject to x in a neighborhood of x^*

ii. $\nabla_x f, \nabla_x g, \nabla_x h$ as well as g, h are continuously differentiable subject to p in a neighborhood of p_0.

Then, a neighborhood $\mathcal{N}_p(p_0)$ of p_0 for continuously differentiable functions $x : \mathcal{N}_p(p_0) \to \mathbb{R}^{n_{opt}}$, $\lambda : \mathcal{N}_p(p_0) \to \mathbb{R}^{n_g}$ and $\mu : \mathcal{N}_p(p_0) \to \mathbb{R}^{n_h}$ such that

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

ii. the active set of a perturbed solution remains the same as the nominal

iii. LICQ holds in $x(p)$

iv. for all $p \in \mathcal{N}_p(p_0)$ the triple $(x(p), \lambda(p), \mu(p))$ fulfills the second-order sufficient optimality conditions with strict complementarity.

Proof. See [30, 19]. □

The sensitivity derivatives can be determined based on the following corollary.

Corollary 3.39 (Sensitivity derivative of optimal solution). *Consider the same assumptions as in Theorem 3.38. The linear equations yield*

$$\begin{pmatrix} \nabla_x^2 L & \nabla_x g^\top & \nabla_x h^\top \\ \nabla_x g & 0 & 0 \\ \operatorname{diag}(\mu) \nabla_x h & 0 & \operatorname{diag}(h) \end{pmatrix} \begin{pmatrix} \frac{\mathrm{d}x}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}\lambda}{\mathrm{d}p}(p_0) \\ \frac{\mathrm{d}\mu}{\mathrm{d}p}(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla_{xp} L \\ \nabla_p g \\ \operatorname{diag}(\mu) \nabla_p h \end{pmatrix} \tag{3.29}$$

the sensitivity derivative of the optimal solution.

Proof. Set $y = (x, \lambda, \mu)$, and use (3.28) to derive from (3.27). □

The derivatives of the objective function and the constraints can also be calculated subject to the perturbation. The objective function is special because its second-order sensitivity derivative can be determined.

Theorem 3.40 (First- and second-order sensitivity derivative of objective under general perturbations). *Consider the same assumptions as in Theorem 3.38. Provided that the objective function f and the constraints g and h are twice continuously differentiable with respect to perturbation p, the first- and second-order sensitivity derivatives of the objective function f are given by*

$$\frac{\mathrm{d}f}{\mathrm{d}p}(p_0) = \nabla_p L$$
$$\frac{\mathrm{d}^2 f}{\mathrm{d}p^2}(p_0) = 2\frac{\mathrm{d}x}{\mathrm{d}p}(p_0)^\top \nabla_{xp} L + \frac{\mathrm{d}x}{\mathrm{d}p}(p_0)^\top \nabla_x^2 L \frac{\mathrm{d}x}{\mathrm{d}p}(p_0) + \nabla_p^2 L$$

Proof. See [19], pp. 84-86. □

From the last theorem, a relation between the sensitivity derivatives of the objective function and the Lagrange multipliers can be derived for constant perturbations.

Corollary 3.41 (First- and second-order sensitivity derivative of objective under constantly perturbed constraints). *Let (x^*, λ^*, μ^*) denote the nominal solution of the perturbed optimization program*

$$\min_x \quad f(x) \tag{3.30a}$$
$$\text{subject to} \quad g_i(x) = q_i,\ i = 1, \ldots, n_g \tag{3.30b}$$
$$h_i(x) \leq q_{i+n_g},\ i = 1, \ldots, n_h \tag{3.30c}$$

with nominal perturbation $q = q_0$. Then, it holds

i. $\frac{\partial}{\partial q_i} f(x^*, q_0) = -\lambda_i^*$ *for* $i = 1, \ldots, n_g$,

ii. $\frac{\partial}{\partial q_{i+n_g}} f(x^*, q_0) = -\mu_i^*$ *for* $i = 1, \ldots, n_h$.

The second-order sensitivity derivatives are given by

i. $\frac{\partial^2}{\partial q_j \partial q_i} f(x^*, q_0) = -\frac{\partial}{\partial q_j}\lambda_i^*$ *for* $i, j = 1, \ldots, n_g$,

ii. $\frac{\partial^2}{\partial q_{j+n_g} \partial q_{i+n_g}} f(x^*, q_0) = -\frac{\partial}{\partial q_{j+n_g}}\mu_i^*$ *for* $i, j = 1, \ldots, n_h$.

Proof. See [19], p. 94. □

With the obtained knowledge about the parametric sensitivity derivatives of the primary and dual optimization variables as well as the objective function, we can estimate the solution $(x(p), \lambda(p), \mu(p))$ and objective value $f(p)$ of the perturbed optimization task. For

this purpose, *first-order Taylor expansions* for the quantities are considered as proposed in [19]:

$$x(p) \approx x(p_0) + \frac{\mathrm{d}}{\mathrm{d}p} x(p_0)(p - p_0) \tag{3.31}$$

$$\lambda(p) \approx \lambda(p_0) + \frac{\mathrm{d}}{\mathrm{d}p} \lambda(p_0)(p - p_0) \tag{3.32}$$

$$\mu(p) \approx \mu(p_0) + \frac{\mathrm{d}}{\mathrm{d}p} \mu(p_0)(p - p_0) \tag{3.33}$$

In the special case of a sufficiently smooth objective function, a second-order approximation is possible:

$$\begin{aligned} f(x(p), p) \approx f(x^*, p_0) + \frac{\mathrm{d}}{\mathrm{d}p} f(x^*, p_0)(p - p_0) \\ + \frac{1}{2}(p - p_0)^\top \frac{\mathrm{d}^2}{\mathrm{d}p^2} f(x^*, p_0)(p - p_0) \end{aligned} \tag{3.34}$$

3.2 Dynamic Optimization

The theory of optimal control arose from the calculus of variations, in which many renowned mathematicians participated (cf. [35]). They were challenged by Johann Bernoulli in 1696 with the so-called *brachistochrone* problem, in which a curve is sought between two points in the vertical plane [55]. On this curve, a (frictionless) ball is supposed to travel from a starting point to an end point in the shortest possible time by gravity alone.
After the Second World War, at the beginning of the Cold War, mathematicians from the East and the West independently developed solution strategies for similar problems in a military context, such as the minimum time interception problems for fighter aircraft [59]. The proof of the maximum principle, which was achieved by the group around Lev Semyonovich Pontryagin in the 1950s [58], finally established the mathematical field of optimal control. In the early 1960s, the field of optimal control blossomed with the arrival of the computer [63]. The latter successfully contributed to the computation of trajectories in the aerospace domain.

3.2.1 Formulation of Optimal Control Problems

One aims to find the optimal procedure to influence typically physical quantities subject to their dynamic behavior and context-dependent restraints by formulating an optimal control problem. The process happens in a time interval $\left[t_0, t_f\right]$. The said quantities are called *states*, while the means to regulate them are termed *controls*. Both are expressed as functions of time:

$$\boldsymbol{x} : \left[t_0, t_f\right] \to \mathbb{R}^{n_x},\ t \mapsto \left(\boldsymbol{x}_1(t), \quad \ldots, \boldsymbol{x}_{n_x}(t)\right)^\top =: \boldsymbol{x}(t) \tag{3.35}$$

$$\text{and } \boldsymbol{u}\colon \left[t_0, t_f\right] \to \mathbb{R}^{n_u},\ t \mapsto \left(\boldsymbol{u}_1(t), \quad \ldots, \boldsymbol{u}_{n_u}(t)\right)^\top =: \boldsymbol{u}(t). \tag{3.36}$$

The state function $\boldsymbol{x}$ shall comply with the *system dynamic* that is a mathematical model based on the real-world behavior of the quantities.

Definition 3.42 (System dynamic). *Let $f_{dyn}\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\chi} \times \left[t_0, t_f\right] \to \mathbb{R}^{n_x}$ be a continuous and partially integrable function with regard to $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$. The first-order ordinary differential equation system*

$$\dot{\boldsymbol{x}}(t) = f_{dyn}(\boldsymbol{x}(t), \boldsymbol{u}(t), \chi, t) \quad \text{for } t \in \left[t_0, t_f\right] \tag{3.37}$$

is called system dynamic. The system dynamic is called autonomous if f_{dyn} does not directly depend on time t. In this case, by redefining $f_{dyn}\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\chi} \to \mathbb{R}^{n_x}$, (3.37) can be stated as

$$\dot{\boldsymbol{x}}(t) = f_{dyn}(\boldsymbol{x}(t), \boldsymbol{u}(t), \chi) \quad \text{for } t \in \left[t_0, t_f\right] \tag{3.38}$$

In the definition of the system dynamic, a time-independent vector χ is included to adjust a more general model. The components in χ constitute the means to configure the optimal control problem overall. We do not need to separately regard the time-independent quantities in the notation in the following because they may be considered states that do not change over time. For this purpose, the state vector and the dynamic system are extended so that (3.37) becomes

$$\begin{pmatrix} \dot{\boldsymbol{x}}(t) \\ \dot{\chi}(t) \end{pmatrix} = \begin{pmatrix} f_{\text{dyn}}(\boldsymbol{x}(t), \boldsymbol{u}(t), \chi, t) \\ 0 \end{pmatrix} \quad \text{for } t \in \left[t_0, t_f\right] \tag{3.39}$$

with $\chi(\cdot) \equiv \chi$. In a similar way, we remove the direct time dependency of f_{dyn} by considering t as a state with $\dot{t} \equiv 1$. We will continue using a notation of an autonomous setting with no time-independent variables.

Definition 3.43 (Boundary conditions). *Let $\Psi\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_\Psi}$ be continuously differentiable with regard to $\boldsymbol{x}(t_0)$ and $\boldsymbol{x}\left(t_f\right)$. The equation*

$$\Psi\left(\boldsymbol{x}(t_0), \boldsymbol{x}\left(t_f\right)\right) = 0 \tag{3.40}$$

is called boundary conditions.

Typically, the states do not reach infinite magnitudes or shall avoid areas of the state space in an application. Furthermore, limits to control capabilities may exist due to resources in the real world. These conditions are incorporated into the optimal control problem as *path constraints*.

Definition 3.44 (Path constraints). *Let $C\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_C}$ be a continuously differentiable function. The componentwise inequalities*

$$C(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq 0 \quad \text{for } t \in \left[t_0, t_f\right] \tag{3.41}$$

are called path constraints.

With Definitions 3.42, 3.43, and 3.44, the feasibility of states $\boldsymbol{x}$ and controls $\boldsymbol{u}$ are specified. It remains to provide a *cost functional* to rate $\boldsymbol{x}$ and $\boldsymbol{u}$.

Definition 3.45 (Cost functional). *Let $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}$ and $l : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ be continuously partially differentiable functions with regard to $\boldsymbol{x}$ and $\boldsymbol{u}$. The term*

$$J[x,u] = f\left(\boldsymbol{x}(t_0), \boldsymbol{x}\left(t_f\right)\right) + \int_{t_0}^{t_f} l(\boldsymbol{x}(t), \boldsymbol{u}(t))\,\mathrm{d}\,t \tag{3.42}$$

is called cost functional.

With the necessary elements set up, an optimal control problem can be defined.

Definition 3.46 (Optimal control problem). *Assume the functions $f_{dyn} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}, \Psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_\Psi}, C : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_C}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}$, and $l : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ are defined as in Definitions 3.42, 3.43, 3.44, and 3.45 with an autonomous system dynamic with no time-independent variables. The minimization task*

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{u}} \quad & f\left(\boldsymbol{x}(t_0), \boldsymbol{x}\left(t_f\right)\right) + \int_{t_0}^{t_f} l(\boldsymbol{x}(t), \boldsymbol{u}(t))\,\mathrm{d}\,t \\ \textit{subject to} \quad & \dot{\boldsymbol{x}}(t) = f_{dyn}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \\ & \Psi\left(\boldsymbol{x}(t_0), \boldsymbol{x}\left(t_f\right)\right) = 0, \\ & C(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq 0, \quad \textit{for } t \in \left[t_0, t_f\right] \end{aligned} \tag{OCP}$$

to find continuous and piecewise continuously differentiable states $\boldsymbol{x} : \left[t_0, t_f\right] \to \mathbb{R}^{n_x}$ and continuous controls $\boldsymbol{u} : \left[t_0, t_f\right] \to \mathbb{R}^{n_u}$ is called optimal control problem.

At first sight, (OCP) seems to be restricted to processes with fixed time intervals only. However, by introducing a new time variable $\tau \in [0,1]$ and the transformation

$$t = t_0 + \tau\left(t_f - t_0\right),$$

problems with free time can be reformulated to have a fixed time frame. The actual interval bounds t_0 and t_f are considered as static variables and are optimized in the solving procedure.
There are three formulations commonly used for (OCP) differing in the terms of the cost functional considered as 0. An OCP is called a

Bolza problem if $f \not\equiv 0$ and $l \not\equiv 0$,

Mayer problem if $f \not\equiv 0$ and $l \equiv 0$,

Lagrange problem if $f \equiv 0$ and $l \not\equiv 0$.

These three forms can be equivalently reformulated into each other. Hereafter, we can make the assumption $f \not\equiv 0$ and $l \equiv 0$ for (OCP) without further restrictions. The original

Bolza problem is transformed into a Mayer problem through the definition of an additional state $z\colon \left[t_0, t_f\right] \to \mathbb{R}$. The system dynamic is supplemented with

$$\dot{z}(t) = l(\boldsymbol{x}(t), \boldsymbol{u}(t))$$

while in the boundary conditions

$$z(t_0) = 0$$

is included. Subsequently, the integral term can be replaced by the summand $z\left(t_f\right)$. A reformulation from a Lagrange to a Mayer problem is performed accordingly. Detailed transformation between the problem types are presented in [66]. In the following, we will assume (OCP) as a Mayer problem because it simplifies the notation and explanation of the transcription of an optimal control problem.

3.2.2 Transcription to Static Optimization Problem

The process of solving an optimal control problem is categorized in *direct* and *indirect methods*. In indirect methods, Euler-Lagrange equations constitute necessary optimality conditions for (OCP), which depends on the definition of the Hamiltonian function and time-dependent adjoint variables. Evaluating these conditions leads to a two-point boundary value problem. Pontryagin's maximum principle states that the control $\boldsymbol{u}$ must be chosen such that the Hamiltonian is optimized. Indirect methods require advanced knowledge of optimal control theory to derive the surrogate problem. Furthermore, the adjoint variables must be estimated, which is not intuitive because they do not represent physical quantities. Therefore, indirect methods are considered rather difficult and not robust [15].
In contrast, direct methods are a general approach to treat (OCP). In the first step, (OCP) is discretized, and then, static optimization techniques are applied as presented in Section 3.1. The discretization procedure is called *transcription*. In the following, we will study the direct approach of solving (OCP) based on *single-step methods* are regarded. For other discretization approaches, it is referred to Section 4.5 in [15] and [62].

Definition 3.47 (Time grid). *Given a time interval $\left[t_0, t_f\right]$ and the desired number of grid points $1 < n_{dis} \in \mathbb{N}$, the set of time points $t_i \in \left[t_0, t_f\right]$, $i = 1, \ldots, n_{dis}$ with*

$$t_0 =: t_1 < t_2 < \cdots < t_{n_{dis}} := t_f$$

is called time grid. The time points in a time grid are not necessarily equidistant. Notationwise,

$$\Delta_i := t_{i+1} - t_i,\ i = 1, \ldots, n_{dis} - 1$$

stands for the respective time increment.

Instead of searching for functions $\boldsymbol{x}$ and $\boldsymbol{u}$, the respective function values at the points of a provided time grid shall be approximated, hence

$$\boldsymbol{x}^{[i]} \approx \boldsymbol{x}(t_i) \text{ and } \boldsymbol{u}^{[i]} \approx \boldsymbol{u}(t_i) \tag{3.43}$$

for $i = 1, \ldots, n_{\text{dis}}$. In this manner, optimization variables

$$x := \begin{pmatrix} \boldsymbol{x}^{[1]} \\ \boldsymbol{u}^{[1]} \\ \vdots \\ \boldsymbol{x}^{[n_{\text{dis}}]} \\ \boldsymbol{u}^{[n_{\text{dis}}]} \end{pmatrix} \in \mathbb{R}^{n_{\text{opt}}}, \quad n_{\text{opt}} := (n_x + n_u) \cdot n_{\text{dis}} \tag{3.44}$$

are allocated in a finite dimensional space. Single-step methods approximate the solution of an ordinary differential equation (or at this point a system dynamic) in the following way:

$$\boldsymbol{x}^{[i+1]} = \boldsymbol{x}^{[i]} + \Delta_i \cdot \Phi\left(\boldsymbol{x}^{[i]}, \boldsymbol{x}^{[i+1]}, \boldsymbol{u}^{[i]}, \boldsymbol{u}^{[i+1]}\right) \tag{3.45}$$

for $i = 1, \ldots, n_{\text{dis}} - 1$. Unlike multi-step methods, function Φ only depends on states and controls at those two time points connected by the time increment Δ_i. The *explicit Euler method*

$$\Phi\left(\boldsymbol{x}^{[i]}, \boldsymbol{x}^{[i+1]}, \boldsymbol{u}^{[i]}, \boldsymbol{u}^{[i+1]}\right) := f_{\text{dyn}}\left(\boldsymbol{x}^{[i]}, \boldsymbol{u}^{[i]}\right)$$

and the implicit *trapezoidal method*

$$\Phi\left(\boldsymbol{x}^{[i]}, \boldsymbol{x}^{[i+1]}, \boldsymbol{u}^{[i]}, \boldsymbol{u}^{[i+1]}\right) := \frac{1}{2}\left(f_{\text{dyn}}\left(\boldsymbol{x}^{[i]}, \boldsymbol{u}^{[i]}\right) + f_{\text{dyn}}\left(\boldsymbol{x}^{[i+1]}, \boldsymbol{u}^{[i+1]}\right)\right)$$

are well known examples in this regard. The approximations (3.43) ought to fulfill the remaining constraints of (OCP) and act as arguments for the cost functional.

Definition 3.48 (Discretized optimal control problem). *Let $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}$, $\Psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_\Psi}$, and $C : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_C}$ denote the cost functional, boundary conditions, and path constraints of* (OCP) *as a Mayer problem, and let Φ be an appropriate single-step expression to approximate its autonomous system dynamic. Furthermore, let $\left\{t_1, \ldots, t_{n_{dis}}\right\}$ denote a time grid for $\left[t_0, t_f\right]$ with time increments $\Delta_1, \ldots, \Delta_{n_{dis}-1}$. The minimization task*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & f\left(\boldsymbol{x}^{[1]}, \boldsymbol{x}^{[n_{dis}]}\right) \\ \text{subject to} \quad & \boldsymbol{x}^{[i+1]} = \boldsymbol{x}^{[i]} + \Delta_i \cdot \Phi\left(\boldsymbol{x}^{[i]}, \boldsymbol{x}^{[i+1]}, \boldsymbol{u}^{[i]}, \boldsymbol{u}^{[i+1]}\right),\ i = 1, \ldots, n_{dis} - 1, \\ & \Psi\left(\boldsymbol{x}^{[1]}, \boldsymbol{x}^{[n_{dis}]}\right) = 0, \\ & C\left(\boldsymbol{x}^{[i]}, \boldsymbol{u}^{[i]}\right) \leq 0,\ i = 1, \ldots, n_{dis} \end{aligned} \tag{3.46}$$

to find $x \in \mathbb{R}^{n_{opt}}$ as defined in (3.44) *is called discretized optimal control problem.*

The static optimization task (3.46) has the same form as (OP), and is said to be transcribed. The number of equality constraints adds up to $n_g = (n_{\text{dis}} - 1) \cdot n_x + n_\Psi$ and $n_h = n_{\text{dis}} \cdot n_C$ is the number of inequalities.

The procedure described in this section is only an example of a transcription of an optimal control problem. Other integration schemes may be applied as well. Thus, the transformation of a Bolza problem into a Mayer problem does not only simplify the illustrations in this section. It is also recommended to perform consistent integration both in the cost functional and while treating the system dynamic.

Chapter 4

Algorithms for Set Approximation

Relying on the foundation we have laid in the previous chapters, we formulate algorithms to approximate geometric bodies in the following. They may illustrate dynamically constrained sets like the reachable set. In the following, $n_{\mathcal{S}} \in \mathbb{N}$ denotes the dimension of some sought-after set $\mathcal{S} \subset \mathbb{R}^{n_{\mathcal{S}}}$. An underlying dynamic feasibility problem, the constraints of an OCP, is assumed to be discretized according to the transcription approach presented in Section 3.2 about dynamic optimization. A feasibility problem is given by a pair of functions (g, h) representing the equalities and inequalities in (OP). As introduced in Definition 3.1, $\mathcal{X}$ denotes the feasible set and is specified through (g, h). With regard to Definition 3.23 of convex programs, the feasibility problem is called *convex* if g is affine and h is convex.
As $x \in \mathcal{X}$ represents states and controls at every time point of the discretization, we are interested in certain components of the optimization vector. In the forward reachability context, we only consider that part of $x \in \mathbb{R}^{n_{\text{opt}}}$ that is assumed to be $\boldsymbol{x}\left(t_f\right) \in \mathbb{R}^{n_x}$, where t_f denotes the final time of the process. The final state vector may be further constrained, or some states are not of interest such that, in turn, only certain components of it need to be taken into account. Therefore, $z \in \mathbb{R}^{n_{\mathcal{S}}}$ is introduced to refer to those components of $x \in \mathbb{R}^{n_{\text{opt}}}$ that are relevant for a given application scenario. The vector z implies a projection

$$Px = z, \tag{4.1}$$

where $P \in \mathbb{R}^{n_{\mathcal{S}} \times n_{\text{opt}}}$ has full rank and standard unit vectors as rows. Accordingly, the relation between $\mathcal{S}$ and the feasible set $\mathcal{X}$ based on (g, h) is expressed through the set evaluation with projection P

$$P\mathcal{X} := \{Px : x \in \mathcal{X}\} = \mathcal{S}.$$

Based on the relation between $\mathcal{X}$ and $\mathcal{S}$, $z \in \mathbb{R}^{n_{\mathcal{S}}}$ is called *feasible* if $z \in \mathcal{S}$.
This chapter consists of four sections. In the first one, simple geometries are introduced as examples to apply the algorithms presented in the following passage. Then, every approach is showcased in a dedicated section. In each, the algorithms are described and illustrated in the first instance. All of them are optimization-based, and depending on the solved task type, different properties may be assigned to the results. These properties are elaborated

on in a detailed and mathematically sound manner. Finally, each section concludes with a discussion showing the example geometries' approximations, and strengths and weaknesses are explained.

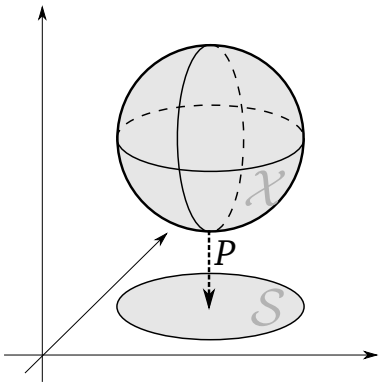


Figure 4.1: Illustration of $P\mathcal{X} = \mathcal{S}$

4.1 Application on Simple Geometries

We first define feasibility problems whose feasible sets represent geometric bodies that serve as application examples. Apart from the dynamic Rayleigh problem, the regarded sets are higher-dimensional generalizations of the ellipse and the rectangular about which properties such as volume are known.

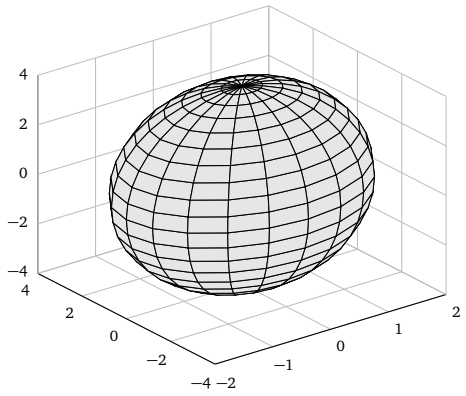


Figure 4.2: A three-dimensional ellipsoid

Definition 4.1 (Static ellipsoid feasibility problem). *Let $n \in \mathbb{N}$ denote the dimension of the set. Furthermore, let $\alpha_1, \ldots, \alpha_n \in \mathbb{R}^+ \setminus \{0\}$ denote positive real scalars. Given the inequality*

$$\sum_{i=1}^{n} \frac{x_i^2}{\alpha_i^2} \leq 1, \tag{E}$$

the set $\mathcal{S}_E(n) := \{x \in \mathbb{R}^n \colon x \text{ complies with } (\mathrm{E})\}$ describes an ellipsoid.

The ellipsoid has a smooth surface and is convex, as illustrated in Figure 4.2 showing a three-dimensional example. To be more precise, an ellipsoid is strictly convex, which means that the line connecting two points in the set lies entirely inside if the two end points are omitted. In other words, the boundary of the ellipsoid does not contain a straight line. The volume of the n-dimensional ellipsoid is

$$\mathrm{vol}(\mathcal{S}_E(n)) = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)} \prod_{i=1}^{n} \alpha_i,$$

where Γ denotes the gamma-function (cf. [17], p. 478).

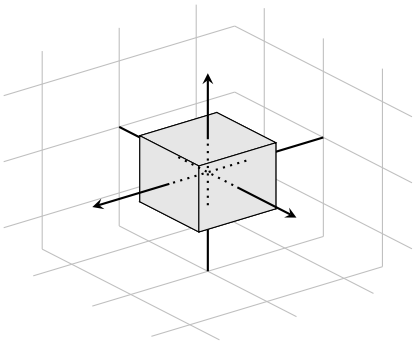


Figure 4.3: A three-dimensional box

Definition 4.2 (Static box feasibility problem). *Let $n \in \mathbb{N}$ denote the dimension of the set. Furthermore, let $\beta_1, \ldots, \beta_n > 0$ denote positive real scalars. Given the inequalities*

$$-\beta_i \leq x_i \leq \beta_i, \quad i = 1, \ldots, n \tag{B}$$

the set $\mathcal{S}_B(n) := \{x \in \mathbb{R}^n : x \text{ complies with } (\mathrm{B})\}$ describes a box.

The box is a convex polytope and has sharp transitions from one facet to another. This can be seen in Figure 4.3, which illustrates a three-dimensional cube. The volume of the n-dimensional box is

$$\mathrm{vol}(\mathcal{S}_B(n)) = 2^n \prod_{i=1}^{n} \beta_i.$$

As a convex polytope, the n-dimensional box is completely defined, when its 2^n corners or vertices are known. $2n$ inequalities suffice to state its $\mathcal{H}$-description.

Definition 4.3 (Dynamic Rayleigh feasibility problem). *Let $[0, 2.5]$ denote the process time interval. For state $\boldsymbol{x} : [0, 2.5] \to \mathbb{R}^2$ and control $\boldsymbol{u} : [0, 2.5] \to \mathbb{R}$,*

$$\begin{aligned}
&\dot{\boldsymbol{x}}(t) = \begin{pmatrix} \boldsymbol{x}_2(t) \\ -\boldsymbol{x}_1(t) + \boldsymbol{x}_2(t)\left(1.4 - 0.14\boldsymbol{x}_2^2\right) + 4\boldsymbol{u}(t) \end{pmatrix}, \\
&\boldsymbol{u}(t) \in [-1, 1], \quad t \in [0, 2.5], \\
&\boldsymbol{x}(0) = \begin{pmatrix} -5 \\ -5 \end{pmatrix}
\end{aligned} \tag{R}$$

describes the Rayleigh problem. The reachable set of the Rayleigh problem is defined as $\mathcal{S}_R := \{x \in \mathbb{R}^2 \colon \exists \boldsymbol{x}, \boldsymbol{u}$ *which complies with* (R) *with* $x = \boldsymbol{x}(2.5)\}$.

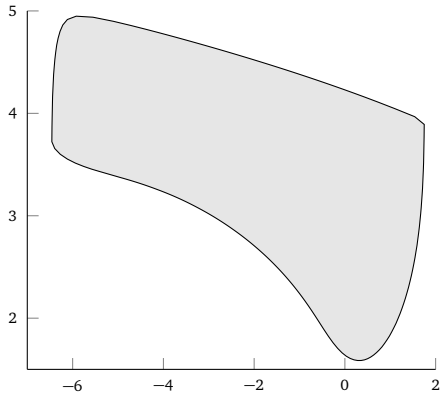


Figure 4.4: Reachable set of the Rayleigh problem

The Rayleigh problem originates from an optimal control problem of an electric circuit [12]. Unlike the other two examples, the sought set of the Rayleigh problem is subject to dynamics. Figure 4.4 shows that this set is not convex.

4.2 Grid Approach

The grid approach is a versatile method for understanding the sought-after set $\mathcal{S}$ that does not need to be convex or even connected. In this approach, a grid is defined over an area where $\mathcal{S}$ is roughly assumed. All grid points are then evaluated whether they are inside the set. This is visualized in Figure 4.5. Some grid points are inside $\mathcal{S}$ and marked blue. For those grid points, which are not inside, the respective closest element in $\mathcal{S}$ is found (orange dots).
To apply this method, we first need to define an *n-dimensional grid*.

Definition 4.4 (*n*-dimensional grid). *Given intervals* $[\gamma_{0,i}, \gamma_{f,i}]$, $i = 1, \ldots, n$, *and the number of discretizations* $1 < n_{dis,i} \in \mathbb{N}$ *for the respective interval, the subsets of discrete points*

$$\Gamma_i := \{\gamma_{1,i}, \gamma_{2,i}, \ldots, \gamma_{n_{dis,i},i}\} \subset [\gamma_{0,i}, \gamma_{f,i}],$$

with increasing values

$$\gamma_{0,i} =: \gamma_{1,i} < \gamma_{2,i} <, \ldots, < \gamma_{n_{dis,i},i} := \gamma_{f,i},$$

defines a discretization of the i-th interval. The set

$$\mathcal{G} := \{(\gamma_1, \ldots, \gamma_n)^\top \in \mathbb{R}^n \colon \gamma_i \in \Gamma_i\} \tag{4.2}$$

is called an n-dimensional grid. $\mathcal{G}$ *consists of* $n_{\mathcal{G}} := \prod_{i=1}^{n} n_{dis,i}$ *elements.*

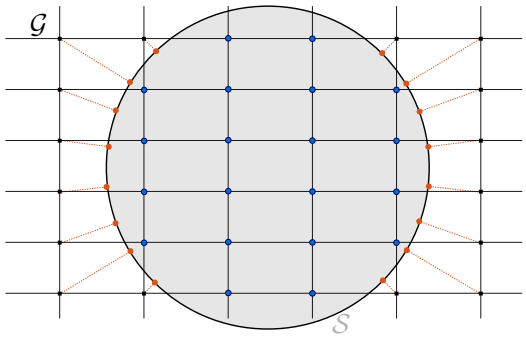


Figure 4.5: Grid $\mathcal{G}$ laid over the area where the sought set $\mathcal{S}$ is

The closest feasible point $z \in \mathcal{S}$ is searched for each grid point. For this purpose, the following optimization problem is solved repeatedly, differing in the targeted grid point.

Definition 4.5 (Grid task). *Let (g,h) denote a feasibility problem. Given a point $p_{\mathcal{G}} \in \mathbb{R}^{n_S}$, the minimization task*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & \frac{1}{2}\|z - p_{\mathcal{G}}\|_2^2 \\ \textit{subject to} \quad & g(x) = 0, \\ & h(x) \leq 0 \end{aligned} \tag{G$(p_{\mathcal{G}})$}$$

with $z = Px$ as in (4.1), is called a grid task.

If the feasibility problem in G$(p_{\mathcal{G}})$ is convex, then G$(p_{\mathcal{G}})$ is a CP. Furthermore, following Example 3.35, we can transform a convex grid task into a SOCP.

Algorithm D Grid Approach

D-1: *(Input)*
- D-1.1: *(Feasibility problem)* Provide feasibility problem (g,h)
- D-1.2: *(Grid)* Provide a suitable n_S-dimensional grid $\mathcal{G}$.

D-2: *(Optimization)* Compute G$(p_{\mathcal{G}})$ for all $p_{\mathcal{G}} \in \mathcal{G}$.

D-3: *(Output)*
- D-3.1: *(Results of optimizations)* From grid task G$(p_{\mathcal{G}})$, $p_{\mathcal{G}} \in \mathcal{G}$
  - Primal variables x^* and point $z^* \in \mathcal{S}$
  - Parametric sensitivities $\nabla_{p_{\mathcal{G}}} x\left(p_{\mathcal{G}}\right)$
- D-3.2: *(Exclusion description)* Set $\mathcal{E} := \left\{\left(z^*, p_{\mathcal{G}}\right) \in \mathcal{S} \times \mathcal{G} : z^* \text{ results from G}\left(p_{\mathcal{G}}\right) \text{ with } \|z^* - p_{\mathcal{G}}\|_2 > 0\right\}$

With these definitions, we may describe the algorithm coherently. The sequence of steps in the grid approach is elaborated in Algorithm D. For a grid point $p_{\mathcal{G}} \in \mathcal{G}$, for which the objective function in G$(p_{\mathcal{G}})$ cannot be minimized to 0, a neighborhood can be specified that does not intersect with $\mathcal{S}$. This is explained in more details in Section 4.2.1. Thus, they represent an output argument in Algorithm D, stated in D-3.2. A byproduct of the optimizations performed in D-2 are the parametric sensitivities. With $\nabla_{p_{\mathcal{G}}} x\left(p_{\mathcal{G}}\right)$ from grid task G$(p_{\mathcal{G}})$ for some $p_{\mathcal{G}}$, first-order information is available that describes how the solution x^* behaves if the grid point $p_{\mathcal{G}}$ is slightly perturbed.

4.2.1 Output Properties

The outcome of the grid approach for feasibility problem (g,h) and an n_S-dimensional grid $\mathcal{G}$ is examined. Based on the results of the optimization in D-3.1, an approximate solution $\hat{x}$ of a grid optimization task $\mathrm{G}(\hat{p})$ for a $\hat{p} \in \mathbb{R}^{n_S}$ can be obtained by

$$\begin{aligned} \hat{x} &= x^* + \nabla_{p_\mathcal{G}} x \left(p_\mathcal{G}\right)^\top \left(\hat{p} - p_\mathcal{G}\right) \\ \text{with} \quad p_\mathcal{G} &= \underset{p \in \mathcal{G}}{\arg\min} \, \|p - \hat{p}\|_2 , \end{aligned} \tag{4.3}$$

where x^* denotes the minimizer of $\mathrm{G}\left(p_\mathcal{G}\right)$. This applies the idea of the Taylor expansion as in (3.31), while the grid serves as a look-up to find the closest expansion point.
The focus in the following is placed on the outcome stated in D-3.2. If the distance between a grid point and the nearest admissible point in the set $\mathcal{S}$ cannot be minimized to 0, there exists a neighborhood around the grid point that does not intersect $\mathcal{S}$.

Lemma 4.6 (Excluded neighborhood). *Let $x^* \in \mathbb{R}^{n_{opt}}$ minimize $G\left(p_\mathcal{G}\right)$ for $p_\mathcal{G} \in \mathbb{R}^{n_S}$. Define $\varepsilon := \|z^* - p_\mathcal{G}\|_2$. Then $\mathring{\mathcal{B}}_\varepsilon\left(p_\mathcal{G}\right) \cap \mathcal{S} = \emptyset$.*

Proof. If $\mathring{\mathcal{B}}_\varepsilon\left(p_\mathcal{G}\right) \cap \mathcal{S} \neq \emptyset$, then $y \in \mathbb{R}^{n_S}$ exists for which $y \in \mathring{\mathcal{B}}_\varepsilon\left(p_\mathcal{G}\right)$ and $y \in \mathcal{S}$ applies. Consequently, $\|y - p_\mathcal{G}\|_2 < \varepsilon$ holds contradicting that x^* minimizes $\mathrm{G}\left(p_\mathcal{G}\right)$. □

Building on Lemma 4.6 and furthermore, assuming that $\mathcal{S}$ is convex, a supporting hyperplane can be determined.

Theorem 4.7 (Supporting hyperplane). *Assume that the sought-after set $\mathcal{S} \subset \mathbb{R}^{n_S}$ is convex and compact. Let $x^* \in \mathbb{R}^{n_{opt}}$ minimize $G\left(p_\mathcal{G}\right)$ for $p_\mathcal{G} \in \mathbb{R}^{n_S}$ with $\varepsilon := \|z^* - p_\mathcal{G}\|_2 > 0$. Then, z^* is a boundary point of $\mathcal{S}$ and*

$$a := \frac{z^* - p_\mathcal{G}}{\|z^* - p_\mathcal{G}\|_2} \tag{4.4}$$

is a normal of a supporting hyperplane to $\mathcal{S}$ at z^* that separates $\mathcal{S}$ from $\mathring{\mathcal{B}}_\varepsilon\left(p_\mathcal{G}\right)$.

Proof. Two statements need to be proved:

i. $a^\top y < a^\top z^*$ for all $y \in \mathring{\mathcal{B}}_\varepsilon\left(p_\mathcal{G}\right)$
ii. $a^\top y \geq a^\top z^*$ for all $y \in \mathcal{S}$

To show i. choose an arbitrary $y \in \mathring{\mathcal{B}}_\varepsilon\left(p_\mathcal{G}\right)$ for which

$$\left\|y - p_\mathcal{G}\right\|_2 < \varepsilon \tag{4.5}$$

holds. The inequality in i. is equivalent to

$$a^\top \left(y - p_\mathcal{G}\right) < a^\top \left(z^* - p_\mathcal{G}\right) \tag{4.6}$$

after subtracting $a^\top p_\mathcal{G}$ from both sides. The relation (4.6) is obtained by applying the Cauchy-Schwarz (C.S.) inequality (cf. [17], p. 31) among others:

$$a^\top \left(y - p_\mathcal{G}\right) \overset{\text{C.S.}}{\leq} \|a\|_2 \left\|y - p_\mathcal{G}\right\|_2 \overset{(4.4)}{=} \left\|y - p_\mathcal{G}\right\|_2 \overset{(4.5)}{<} \varepsilon$$

$$\varepsilon = \|z^* - p_{\mathcal{G}}\|_2 = \frac{\left(z^* - p_{\mathcal{G}}\right)^\top \left(z^* - p_{\mathcal{G}}\right)}{\|z^* - p_{\mathcal{G}}\|_2} \overset{(4.4)}{=} a^\top \left(z^* - p_{\mathcal{G}}\right)$$

For ii., assume the opposite. Suppose there exists a $y \in \mathcal{S}$ such that $a^\top y < a^\top z^*$ holds, implying $y \neq z^*$. Then equivalently,

$$\varepsilon a^\top (y - z^*) < 0 \tag{4.7}$$

applies. Because of Lemma 4.6, $y \notin \mathring{\mathcal{B}}_\varepsilon\left(p_{\mathcal{G}}\right)$ and thus,

$$\left\|y - p_{\mathcal{G}}\right\|_2 \geq \varepsilon \tag{4.8}$$

hold. Since $\mathcal{S}$ is convex, $\bar{y} := z^* + \alpha(y - z^*)$ with $\alpha := -\frac{\varepsilon a^\top (y - z^*)}{\|y - z^*\|_2^2}$ is an element of $\mathcal{S}$ if $0 \leq \alpha \leq 1$. However,

$$\begin{aligned}
\left\|\bar{y} - p_{\mathcal{G}}\right\|_2^2 &= \left\|z^* - p_{\mathcal{G}} + \alpha(y - z^*)\right\|_2^2 \\
&= \left\|z^* - p_{\mathcal{G}}\right\|_2^2 + 2\alpha\left(z^* - p_{\mathcal{G}}\right)^\top (y - z^*) + \alpha^2 \|y - z^*\|_2^2 \\
&= \varepsilon^2 - 2\varepsilon^2 \frac{\left(a^\top (y - z^*)\right)^2}{\|y - z^*\|_2^2} + \varepsilon^2 \frac{\left(a^\top (y - z^*)\right)^2 \|y - z^*\|_2^2}{\|y - z^*\|_2^4} \\
&= \varepsilon^2 - \varepsilon^2 \frac{\left(a^\top (y - z^*)\right)^2}{\|y - z^*\|_2^2} < \varepsilon^2
\end{aligned} \tag{4.9}$$

and, consequently, $\left\|\bar{y} - p_{\mathcal{G}}\right\| < \varepsilon$ hold contradicting Lemma 4.6.
It remains to show $0 \leq \alpha \leq 1$. From (4.7), $\alpha > 0$ follows directly. On the other hand, $\alpha \leq 1$ is equivalent to

$$-\varepsilon a^\top (y - z^*) - \|y - z^*\|_2^2 \leq 0. \tag{4.10}$$

Insert the definitions of a and ε into the left hand side of (4.10) and derive

$$\begin{aligned}
&\left(p_{\mathcal{G}} - z^*\right)^\top (y - z^*) - (y - z^*)^\top (y - z^*) \\
&= \left(p_{\mathcal{G}} - y\right)^\top (y - z^*) \\
&= \left(p_{\mathcal{G}} - y\right)^\top \left(y - p_{\mathcal{G}} + p_{\mathcal{G}} - z^*\right) \\
&= \left(p_{\mathcal{G}} - y\right)^\top \left(p_{\mathcal{G}} - z^*\right) - \left\|y - p_{\mathcal{G}}\right\|_2^2 \\
&\overset{\text{C.S.}}{\leq} \left\|p_{\mathcal{G}} - y\right\| \left\|p_{\mathcal{G}} - z^*\right\| - \left\|y - p_{\mathcal{G}}\right\|_2^2 = \varepsilon \left\|y - p_{\mathcal{G}}\right\| - \left\|y - p_{\mathcal{G}}\right\|_2^2 \overset{(4.8)}{\leq} 0
\end{aligned}$$

Thus, (4.10) and, consequently, $\alpha \leq 1$ are fulfilled. □

Summarizing, in case the feasibility problem (g, h) leads to a convex $\mathcal{S}$, and Algorithm D yields a non-empty $\mathcal{E}$, a superset of $\mathcal{S}$ is given by

$$\left\{ y \in \mathbb{R}^{n_S} : \left(\frac{z^* - p_{\mathcal{G}}}{\|z^* - p_{\mathcal{G}}\|_2}\right)^\top y \geq \left(\frac{z^* - p_{\mathcal{G}}}{\|z^* - p_{\mathcal{G}}\|_2}\right)^\top z^*, \text{ for all } \left(z^*, p_{\mathcal{G}}\right) \in \mathcal{E} \right\} \supseteq \mathcal{S} \tag{4.11}$$

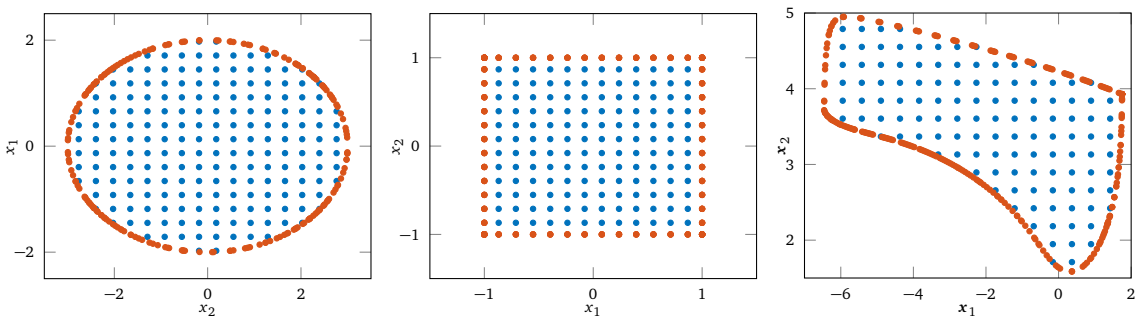
applying Theorem 4.7. Independent of the shape of $\mathcal{S}$,

$$\bigcup_{(z^*,p_\mathcal{G})\in\mathcal{E}} \mathring{\mathcal{B}}_{\|z^*-p_\mathcal{G}\|_2}(p_\mathcal{G}) \cap \mathcal{S} = \emptyset$$

holds because of Lemma 4.6.

4.2.2 Discussion

The greatest strength of the grid approach is that it can be used on any feasibility problem (g,h) as long as the feasibility set and consequently $\mathcal{S}$ is not empty. The versatility is evident in the plots of Figure 4.6, illustrating the results for the examples in Section 4.1. Regardless of the shape of the set to be reconstructed, it can be represented discretely by a point cloud.



(a) Ellipse (E) with $\alpha_1 = 2$, $\alpha_2 = 3$ **(b)** Square (B) with $\beta_1 = \beta_2 = 1$ **(c)** Rayleigh problem (R)

Figure 4.6: Reconstruction of example sets using the grid approach

The approximate location and extent of $\mathcal{S}$ must be estimated in advance to design and provide a suitable grid as input for Algorithm D. The fewer details are known, the smaller the area of $\mathcal{S}$ that the resulting point cloud can cover. A preliminary analysis is therefore essential. In the process, the appropriate density of the grid points also has to be assessed. The density correlates with the number of optimizations. The largest computational effort is due to the optimizations in D-2. However, they can be performed in parallel because there are no dependencies among them. In the grid task $\mathrm{G}(p_\mathcal{G})$, there are no additional constraints besides the feasibility problem (g,h) because the grid point $p_\mathcal{G}$ is integrated into the objective function. The grid optimization task can be reformulated as an SOCP but this comes with an additional optimization variable and $n_\mathcal{S}+1$-dimensional second-order cone constraint.

Since similar problems are solved in D-2, a *warm start* of the solver may be considered. During a warm start, feasible variables are provided from the previous optimization run. Furthermore, no new memory allocation is needed because problem specific details like the sparsity remains unchanged. In this manner, convergence of NLPs are achieved faster in practice. It is advisable to choose the next grid point in the immediate vicinity of the last one for a re-parametrization of the grid task of a solver thread in order to benefit from a

superlinear or quadratic convergence rate through a warm start in the best case (compare Theorem 3.21).
If $\mathcal{S}$ is convex, a non-empty output argument $\mathcal{E}$ from D-3.2 implies an over-approximation of the sought set $\mathcal{S}$ defined in (4.11). This superset can be used to rule out elements in the $\mathbb{R}^{n_S}$ that are certainly not in $\mathcal{S}$. Generally, in order to check whether an element $\hat{p} \in \mathbb{R}^{n_S}$ is in $\mathcal{S}$, it cannot be avoided to solve $\mathrm{G}(\hat{p})$. There is the option here to use the approximation $\hat{x}$ defined in (4.3) as an initial guess for the optimization to aim for a good convergence behavior. Regarding $\mathcal{S}$ in a dynamic context, the respective minimizer x^* of $\mathrm{G}\big(p_{\mathcal{G}}\big)$ for each $p_{\mathcal{G}} \in \mathcal{G}$ represents a trajectory and a control. The output stated in D-3.1 therefore forms a list of theoretically actionable options with ready trajectories and controls.
Appropriate computational resources must be made available to obtain a dense point cloud. This is especially necessary for higher dimensional sets because with uniform axis discretization, the number of grid points increases exponentially with each dimension.

4.3 Online Convex Hull Approach

The online convex hull approach approximates convex sets $\mathcal{S}$ through two polytopes. Both approach the shape of the sought-after set over time. Successively, new boundary points of $\mathcal{S}$ are determined, which are included in the set of vertices of one polytope $\mathcal{P}_{\text{inner}}$. The other, $\mathcal{P}_{\text{outer}}$, is specified by the halfspaces that respectively contain $\mathcal{S}$ and are given through the supporting hyperplanes at the boundary points. Throughout the process, the relation $\mathcal{P}_{\text{inner}} \subseteq \mathcal{S} \subseteq \mathcal{P}_{\text{outer}}$ holds. The course of the algorithm is hinted at in Figure 4.7. While the inner polytope grows with each new boundary point, the outer one shrinks because the intersection of more and more halfspaces is formed. The sequential progression of adding vertices to $\mathcal{P}_{\text{inner}}$ or intersecting halfspaces of $\mathcal{P}_{\text{outer}}$ is illustrated with fading colors.

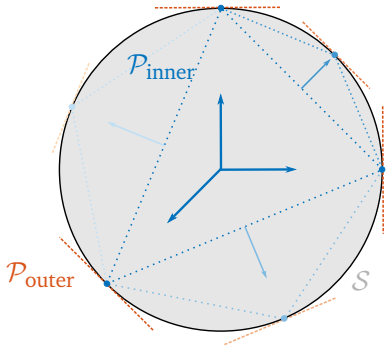


Figure 4.7: The online convex hull's subset and superset of $\mathcal{S}$

The boundary points, which significantly influence the shape of both polytopes, are obtained by solving optimization tasks. In an optimization run, an element $z \in \mathcal{S}$ is determined, which is the highest point above a hyperplane that goes through the origin. The hyperplane and also the definition of *above* is given through a vector $d \in \mathbb{R}^{n_S}$.

Definition 4.8 (Online convex hull task). *Let (g,h) denote a feasibility problem. Given a direction $d \in \mathbb{R}^{n_S}$, the minimization task*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}} \quad & -d^\top z \\ \text{subject to} \quad & g(x) = 0 \\ & h(x) \leq 0 \end{aligned} \tag{OCH(d))}$$

with $z = Px$ as in (4.1), is called an online convex hull task.

The objective of an online convex hull task is linear. Provided a convex feasibility problem, OCH(d) represents a CP.

Algorithm E Online Convex Hull Approach

E-1: *(Input)*
- E-1.1: *(Feasibility problem)* Provide feasibility problem (g,h).
- E-1.2: *(Termination parameter)* Provide termination parameter
  - Maximum number $n_{\texttt{stop}} \geq n_S + 1$ of optimizations to be performed
  - Threshold $\varepsilon_{\texttt{vol}} > 0$ for difference of volumes

E-2: *(Setup)* Define $k := n_S + 1$
- E-2.1: *(Initial vertices)* Obtain initial vertices $\{z^{*,1}, \ldots, z^{*,k}\}$ from solving OCH(d^i), $i = 1, \ldots, k$ with

$$d^i = \begin{cases} e^i, & \text{for } i = 1, \ldots, k-1 \\ -\left(\frac{1}{\sqrt{n_S}}, \ldots, \frac{1}{\sqrt{n_S}}\right)^\top, & \text{for } i = k \end{cases} \tag{4.12}$$

- E-2.2: *(Inner polytope)* Create inner polytope $\mathcal{P}^k_{\text{inner}} := \text{conv}(\{z^{*,1}, \ldots, z^{*,k}\})$
- E-2.3: *(Outer polytope)* Create outer polytope $\mathcal{P}^k_{\text{outer}} := \{y \in \mathbb{R}^{n_S} : A^k y \leq b^k\}$ where

$$A^k_i = \left(d^i\right)^\top, \quad b^k_i = \left(d^i\right)^\top z^{*,i} \tag{4.13}$$

  are the respective i-th row of matrix $A^k \in \mathbb{R}^{n_k \times n_S}$ and vector $b^k \in \mathbb{R}^{n_k}$.

E-3: *(Termination Criteria)* If $k \geq n_{\texttt{stop}}$ or $\text{vol}(\mathcal{P}^k_{\text{outer}}) - \text{vol}(\mathcal{P}^k_{\text{inner}}) < \varepsilon_{\texttt{vol}}$ is true, go to step E-8.

E-4: *(Search direction)* Determine the outer unit normal vector d^{k+1} that is orthogonal to a facet $\mathcal{F}$ of $\mathcal{P}^k_{\text{inner}}$.

E-5: *(Optimization)* Obtain vertex $z^{*,k+1}$ from solving OCH(d^{k+1})

E-6: *(Update)*
- E-6.1: *(Inner polytope)* $\mathcal{P}^{k+1}_{\text{inner}} := \text{conv}(\mathcal{P}^k_{\text{inner}} \bigcup \{z^{*,k+1}\})$ (use Algorithm F)
- E-6.2: *(Outer polytope)* $\mathcal{P}^{k+1}_{\text{outer}} := \{y \in \mathbb{R}^{n_S} : A^{k+1} y \leq b^{k+1}\}$ with

$$A^{k+1} := \begin{bmatrix} A^k \\ \left(d^{k+1}\right)^\top \end{bmatrix}, \quad b^{k+1} = \begin{pmatrix} b^k \\ \left(d^{k+1}\right)^\top z^{*,k+1} \end{pmatrix} \tag{4.14}$$

E-7: *(Increment and repeat)* Set $k := k + 1$, go to step E-3.

E-8: *(Output)*
- E-8.1: *(Results of optimizations)* Minimizer $x^{*,i}$ of online convex hull task OCH(d^i), $i = 1, \ldots, k$
- E-8.2: *(Inner polytope)* Under-approximation $\mathcal{P}_{\text{inner}} := \mathcal{P}^k_{\text{inner}}$
- E-8.3: *(Outer polytope)* Over-approximation $\mathcal{P}_{\text{outer}} := \mathcal{P}^k_{\text{outer}}$

The online convex hull algorithm is described in Algorithm E. Given a feasibility problem and a parametrization for the algorithm, in the setup step E-2, $n_S + 1$ optimizations are performed with search direction $e^i \in \mathbb{R}^{n_S}$, i.e. i-th standard unit vector, yielding a corresponding number of vertices. Based on these, an initial inner and outer polytope are

created. While the inner polytope is the convex hull of the vertices, the outer polytope is the intersection of the halfspaces, the hyperplanes at the vertices yield. Steps E-3 to E-7 are carried out in a loop. Depending on the current state of the inner and outer polytope, optimizations are performed to obtain new vertices and to update the two polytopes. This loop is exited when the difference in volume of both inner and outer polytope falls below the threshold $\varepsilon_{\texttt{vol}}$, or the number of optimizations runs exceeds the limit $n_{\texttt{stop}}$. Both termination parameters are considered as inputs in E-1.2. Assuming the initial vertices $z^{*,1},\dots,z^{*,n_S+1}$ are affinely independent, the inner polytope in E-2.2 is an n_S-simplex. The n_S+1 facets of the simplex are directly identified, since the convex hull of every combination of n_S out of the n_S+1 vertices is a facet. The outer normal vectors of the facets of $\mathcal{P}^k_{\text{inner}}$ represent parametrizations of potential optimization runs OCH(d). It is therefore important to keep track of them, which is realized through a set $\mathcal{I}^k$ of ordered identifiers. Every time a new vertex $z^{*,k+1}$ is determined, both polytopes need to be updated in E-6. The new outer polytope $\mathcal{P}^{k+1}_{\text{outer}}$ is obtained by appending a row to A^k and b^k as done in (4.14). The update routine necessary in E-6.1 is described in Algorithm F. The output of Algorithm E is eventually given by the solutions of the optimization runs OCH$\left(d^i\right)$, $i=1,\dots,k$ and the final inner and outer polytope $\mathcal{P}_{\text{inner}}$ and $\mathcal{P}_{\text{outer}}$.

Algorithm F Incremental Convex Hull

F-1: *(Input)* Expects $k > n_S+1$
- F-1.1: *(Vertices)* Provide vertices $\left\{z^1,\dots,z^k\right\}$
- F-1.2: *(Ordered Identifiers)* Provide set of ordered identifiers $\mathcal{I}^k$ for facets of $\mathcal{P}^k := \text{conv}\left(\left\{z^1,\dots,z^k\right\}\right)$
- F-1.3: *(New vertex)* Provide new vertex z^{k+1}

F-2: *(Setup)* Define
- F-2.1: *(Interior point)* $o := \frac{1}{k+1}\sum_{i=1}^{k+1} z^i$
- F-2.2: *(Shifted vertices)* $v^i = z^i - o$, $i=1,\dots,k+1$
- F-2.3: *(Normal vector)* Given an ordered identifier $I \in \{1,\dots,k+1\}^{n_S}$, define $\vec{n}_I \in \mathbb{R}^{n_S}$ with

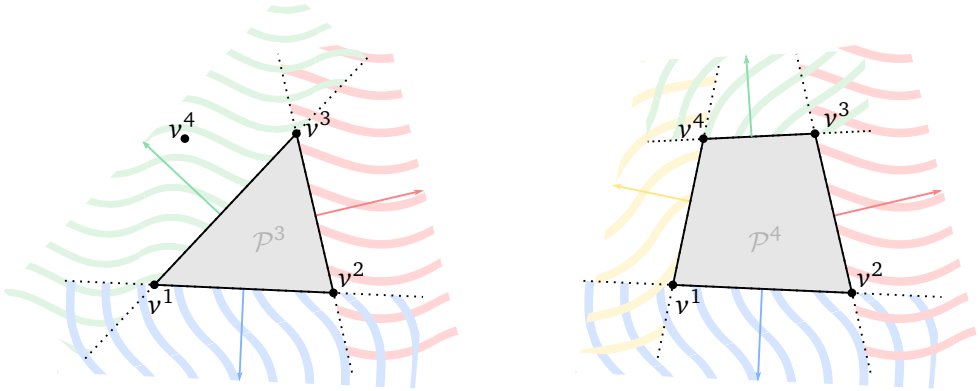
$$V_I \vec{n}_I = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad \text{where } V_I = \begin{bmatrix} \left(v^{I_1}\right)^\top \\ \vdots \\ \left(v^{I_{n_S}}\right)^\top \end{bmatrix}.$$

F-3: *(Loop)* For all $I \in \mathcal{I}^k$, check if $\left(v^{k+1}\right)^\top \vec{n}(I) \le 1$:
- F-3.1: *(Conserve)* If condition holds, insert I into $\mathcal{I}^{k+1}$.
- F-3.2: *(Replace)* Otherwise, go through all $\binom{n_S}{n_S+1}$ combinations of elements in $\left\{I_1,\dots,I_{n_S},k+1\right\}$. Insert combination $\tilde{I}$ as an ordered identifier into $\mathcal{I}^{k+1}$ for which these conditions hold:
  - i. $V_{\tilde{I}}$ has full rank
  - ii. $v^\top \vec{n}\left(\tilde{I}\right) \le 1$ for all $v \in \left\{v^1,\dots,v^{k+1}\right\}$

F-4: *(Output)* Updated set of ordered identifiers $\mathcal{I}^{k+1}$ for facets of $\mathcal{P}^{k+1} := \text{conv}\left(\mathcal{P}^k \bigcup \left\{z^{*,k+1}\right\}\right)$

Based on the ordered identifiers $\mathcal{I}^k$ of the facets of $\mathcal{P}^k_{\text{inner}} = \text{conv}\left(\left\{z^{*,1},\dots,z^{*,k}\right\}\right)$, Algorithm F is the method to determine $\mathcal{I}^{k+1}$ for the facets of $\mathcal{P}^{k+1}_{\text{inner}} = \text{conv}\left(\left\{z^{*,1},\dots,z^{*,k+1}\right\}\right)$. The facets of $\mathcal{P}^k_{\text{inner}}$ must be checked if they are still prevailing. In other words, it is inspected with the outer normal vectors whether the newly determined vertex $z^{*,k+1}$ is above or under a facet. For those facets, the latter can be confirmed for, $\mathcal{I}^{k+1}$ inherits the corresponding ordered identifier from $\mathcal{I}^k$. Otherwise, new identifiers are created by connecting the ridges

of the outdated facets with $z^{*,k+1}$. If a new identifier gives a facet, it is pushed into $\mathcal{I}^{k+1}$. In this process, shifted vertices are defined in F-2.2 based on an interior point $o \in \mathbb{R}^{n_S}$ such that the resulting normal vector $\vec{n}_I$ as in F-2.3 points outwards. Figure 4.8 illustrates how the incremental hull is performed for a two-dimensional example where a fourth vertex v^4 shall be added to the set of vertices $\{v^1, v^2, v^3\}$ of polytope $\mathcal{P}^3$. The new point v^4 is beneath the facets $\text{conv}(\{v^1, v^2\})$ and $\text{conv}(\{v^2, v^3\})$ corresponding to ordered identifiers $(1,2)$ and $(2,3)$. Therefore, these elements prevail in $\mathcal{P}^4$ and $\mathcal{I}^4$ respectively. However, v^4 is above the facet with the ordered identifier $(1,3)$. For all combinations of indices of $\{1,3,4\}$, it is checked whether they represent facets of $\mathcal{P}^4$. $(1,3)$ has already been excluded, so $(1,4)$ and $(3,4)$ remain to be checked. All regarded vertices are respectively beneath $\text{conv}(\{v^1, v^4\})$ and $\text{conv}(\{v^3, v^4\})$, which confirms that they are facets of $\mathcal{P}^4$. The respective ordered identifier can then be inserted into $\mathcal{I}^4$.



(a) $\mathcal{I}^3 = \{(1,2),(2,3),(1,3)\}$ **(b)** $\mathcal{I}^4 = \{(1,2),(2,3),(3,4),(1,4)\}$

Figure 4.8: Illustration of the incremental convex hull

4.3.1 Output Properties

The results of the online convex hull approach can be further processed in a variety of ways. In case the feasibility problem (g,h) is convex, an element of the feasibility set can be specified from a given point in the inner polytope.

Proposition 4.9. *Assume Algorithm E has been performed with $k \leq n_{stop}$ optimization runs, given a convex feasibility problem (g,h) and some termination parameters as input. For any $z \in \mathcal{P}_{inner}$, the solution of*

$$\begin{bmatrix} z^{*,1} & \dots & z^{*,k} \\ 1 & \dots & 1 \end{bmatrix} c = \begin{pmatrix} z \\ 1 \end{pmatrix} \quad \text{with } c_i \geq 0,\ i = 1,\dots,k. \tag{4.15}$$

yields a feasible $x \in \mathbb{R}^{n_{opt}}$ with

$$x = c_1 x^{*,1} + \dots + c_k x^{*,k} \tag{4.16}$$

and $Px = z$.

Proof. As $z \in \mathcal{P}_{\text{inner}}$ is an element of a convex hull, a vector of coefficients $c \in \mathbb{R}^k$ exists, such that $z = c_1 z^{*,1} + \cdots + c_k z^{*,k}$ is a convex combination. The components of c are positive and add up to 1. The solution of (4.15) provides suitable coefficients. The feasibility problem is convex and so is the feasibility set due to Lemma 3.25. Thus, the convex combination (4.16) is in the feasibility set (Lemma 2.4). Furthermore, because of

$$Px = c_1 P x^{*,1} + \cdots + c_k P x^{*,k} = c_1 z^{*,1} + \cdots + c_k z^{*,k} = z,$$

x is projected onto z by P. □

Since the facets of the inner polytope $\mathcal{P}_{\text{inner}}$ are known because the boundary description is updated in every loop cycle in E-6.1, the $\mathcal{H}$-representation can be specified. With that, a candidate $y \in \mathbb{R}^{n_S}$ can be checked if it is an element of $\mathcal{P}_{\text{inner}}$, which implies $y \in \mathcal{S}$. The converse does not necessarily hold.
The inequalities in the definition of the outer polytope in E-2.3 and E-6.2 are established by the following theorem.

Theorem 4.10 (Supporting hyperplane). *Let $x^* \in \mathbb{R}^{n_{opt}}$ minimize OCH(d) for $d \in \mathbb{R}^{n_S}$. Then, d is the normal vector of a supporting hyperplane to $\mathcal{S}$ at z^*. Furthermore, $\mathcal{S}$ is contained in the corresponding halfspace:*

$$\mathcal{S} \subset \left\{ y \in \mathbb{R}^{n_S} : d^\top y \leq d^\top z^* \right\}. \tag{4.17}$$

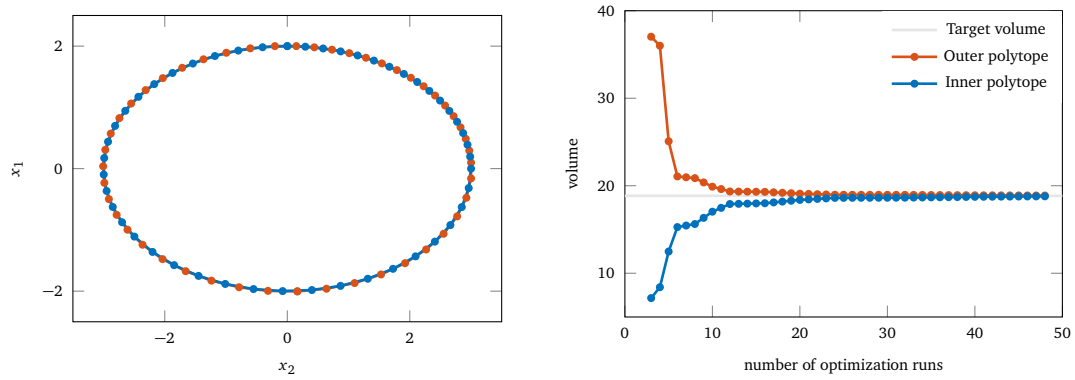
Proof. Suppose, $\mathcal{S}$ is not fully contained in the halfspace, i.e. $y \in \mathcal{S}$ exists for which $d^\top y > d^\top z^*$ holds. Then, equivalently $-d^\top y < -d^\top z^*$ holds and x^* is not the minimizer of OCH$\left(d^i\right)$. □

The outer polytope $\mathcal{P}_{\text{outer}}$ is therefore an over-approximation of $\mathcal{S}$ given in the $\mathcal{H}$-representation. Thus, for all $y \notin \mathcal{P}_{\text{outer}}$, it follows $y \notin \mathcal{S}$

4.3.2 Discussion

The online convex hull approach makes use of the convexity of the set $\mathcal{S}$ and successively approximates the latter with one polytope from the inside and another one from the outside. This can be observed in Figure 4.9, which illustrates the results when approximating an ellipse. This ellipse is defined by (E) with $\alpha_1 = 2$ and $\alpha_2 = 3$. After 48 optimizations, $\mathcal{P}_{\text{inner}}$ (blue) and $\mathcal{P}_{\text{outer}}$ (orange) cover almost the same area in Figure 4.9a. The convergence of the volumes of the inner and outer polytopes towards each other with increasing number of optimizations is illustrated in Figure 4.9b. The inner polytope covers 90% of the volume of the outer polytope after 10 optimization runs.
In Figure 4.10, the same behavior can be observed in the case of the ellipsoid specified by (E) with $\alpha_1 = 2$, $\alpha_2 = 3$ and $\alpha_3 = 4$. The inner approximations after 16, 64, and 256 optimizations are showcased, and the ellipsoid form becomes clearer at each stage. The outer approximations are neglected in these figures to ensure a clean visualization. It takes visibly more optimizations to close the gap between the inner and outer polytope compared

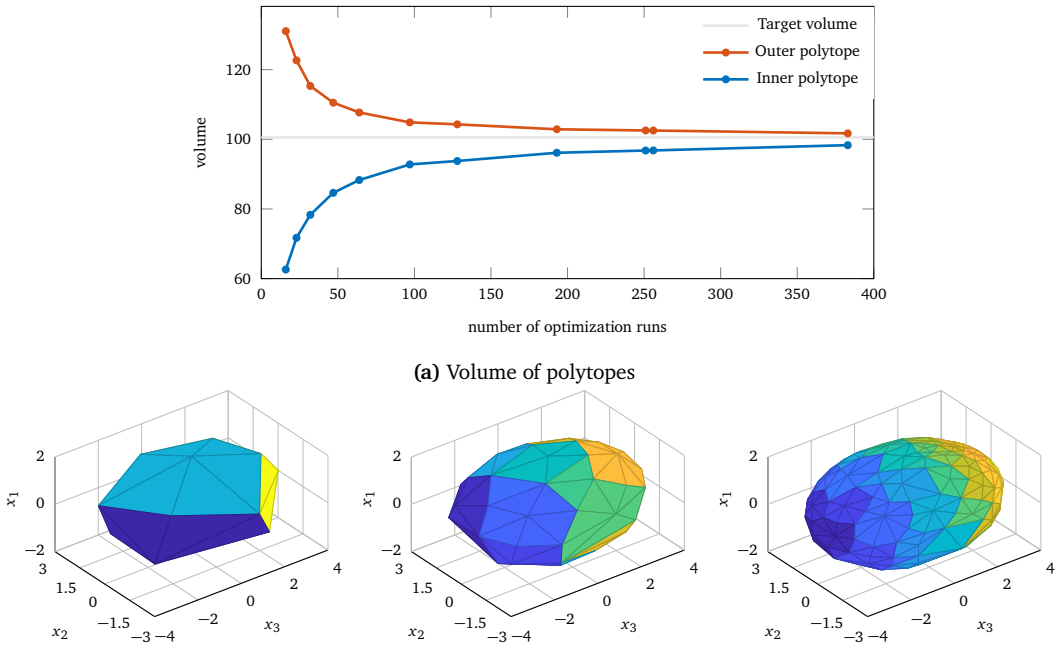
to the 2D example which should be taken into account when choosing the termination parameter. The inner polytope achieves 90% of the outer polytope after 130 optimization runs.



(a) After 48 optimizations

(b) Volume of polytopes

Figure 4.9: Approximation of an ellipse using the online convex hull approach



(a) Volume of polytopes

(b) After 16 optimizations

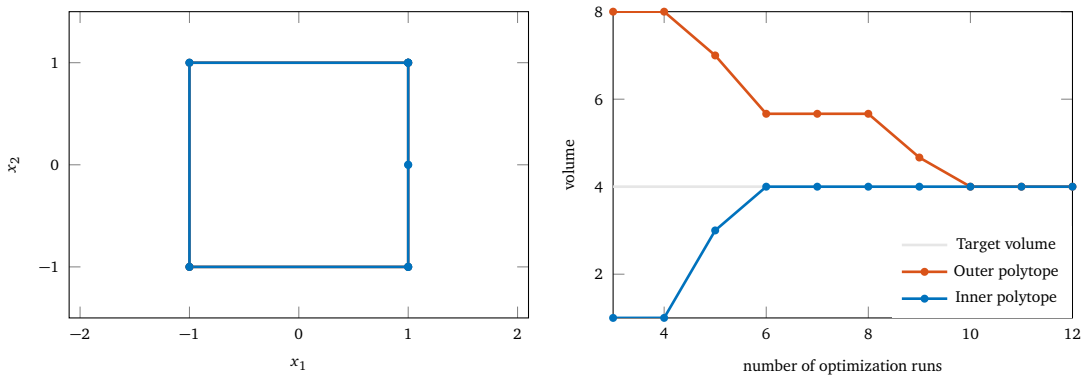
(c) After 64 optimizations

(d) After 256 optimizations

Figure 4.10: Approximation of an ellipsoid using the online convex hull approach

The approximation results of the square are shown in Figure 4.11. The square, specified by (B) with $\beta_1 = \beta_2 = 1$, can be completely reconstructed. While theoretically, only four optimizations are needed in the best case, the volume of the inner polytope reaches the actual volume of the square after six optimizations. The reason for this is the initial

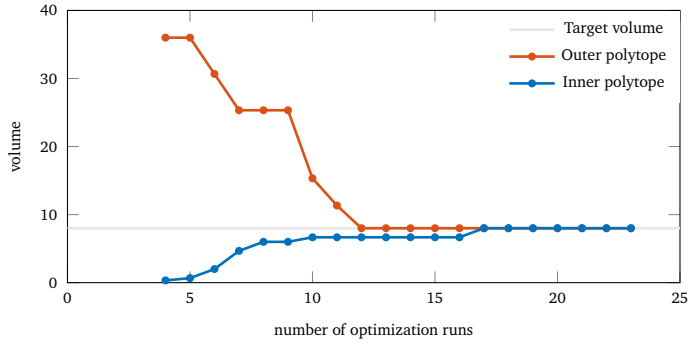
search directions in E-2.1, as well as the non-prioritized facet selection, and thus, direction parametrization in E-4. After four optimization runs, the four corners of the searched square are not necessarily found. After a total of ten optimizations, the reconstruction is secured with the outer polytope covering the same area as the inner polytope. The unsurprising conclusion that polytopes best approximate a convex polytope is thus underlined by this example.



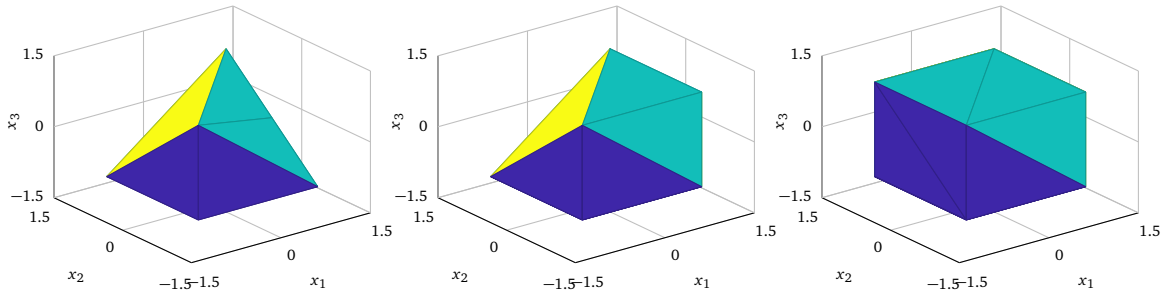
(a) After 12 optimizations

(b) Volume of polytopes

Figure 4.11: Approximation of a square using the online convex hull approach



(a) Volume of polytopes



(b) After 7 optimizations

(c) After 14 optimizations

(d) After 17 optimizations

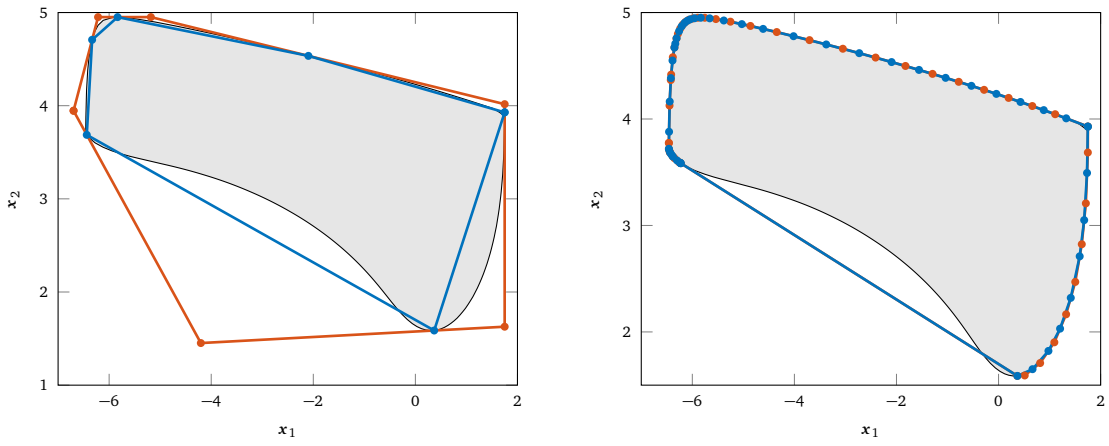
Figure 4.12: Approximation of a cube using the online convex hull approach

Figure 4.12 depicts the approximation results of the three-dimensional box, given by (B) with $\beta_1 = \beta_2 = \beta_3 = 1$. Also here, the sought set can be completely reconstructed. This is achieved after 17 optimizations. The outer polytope is congruent with the cube after 12 optimizations, while the growth of the inner polytope stagnates in the meantime. The latter can be traced back to the fact that if a facet at some stage of the approximation is a subset of the boundary of the sought set, no growth in the outer normal direction is achieved.
On closer inspection, Figure 4.11a shows that the mapping of a search direction d onto a boundary point $z^* \in \partial\mathcal{S}$, as implied by OCH($d$), is not injective. This can lead to considerable startup difficulties after the setup step E-2. In E-2.1, the initial vertices $z^{*,i}$, $i = 1, \ldots, n_\mathcal{S}$ resulting from OCH$\left(e^i\right)$ for an $n_\mathcal{S}$-dimensional box could all lead to the same result

$$z^{*,i} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{n_\mathcal{S}} \end{bmatrix}.$$

Together with the vertex $z^{*,n_\mathcal{S}+1}$, the initial inner polytope is just a line segment and not an $n_\mathcal{S}$-simplex. Thus, further search directions in step E-4 cannot be uniquely specified for $n_\mathcal{S} > 2$. In [28], an alternative setup of the initial polytope is described. In one optimization problem, $n_\mathcal{S} + 1$ vertices are searched on the boundary $\mathcal{S}$ such that the volume of a resulting symmetric simplex is maximal. In the objective function, a determinant is maximized according to the volume formula (2.6). If the underlying feasibility problem is convex, semi-definite programming techniques can be used to solve this problem. As appealing as it is to perform only one optimization instead of $n_\mathcal{S} + 1$, this one optimization is correspondingly $n_\mathcal{S} + 1$ times as large as OCH(d) because the feasibility problem must be adhered to by all vertices respectively. For high-dimensional optimization tasks such as transcribed optimal control problems, exploiting the sparsity patterns of internal matrices is mandatory especially for this application.
It is necessary to calculate the volumes of the inner and outer polytopes to check the termination criterion in E-3. The initial values result from the formula (2.6). All facets of $\mathcal{P}^k_{\text{inner}}$ that, according to the check in F-3, are no longer facets of $\mathcal{P}^{k+1}_{\text{inner}}$ due to a new vertex $z^{*,k+1}$, serve as the bases of simplices that complete $\mathcal{P}^k_{\text{inner}}$ to $\mathcal{P}^{k+1}_{\text{inner}}$. Therefore, instead of computing the volume of the entire inner polytope, it is sufficient to compute the volume increment after each iteration. The procedure is similar for the outer polytope. With each optimization, an additional halfspace potentially cuts off some of the volume. It is sufficient to determine the volume reduction here. However, only in two dimensions this cutoff is easy to determine, because in this case it is a triangle. In higher dimensions, it is a polytope, which, in turn, must be covered with simplices, whose volumes can be specified.
The largest computational effort is also found in the optimizations in E-5 in this approach. The online convex hull optimization task does not impose any constraints other than the feasibility problem and has a linear objective function. Feasible warm starts without reallocating memory are therefore also possible in this approach because the optimization tasks do not differ structurally. A prioritization strategy for the selected facet in E-4, which determines the next optimization, is a potentially useful addition to Algorithm E in order to make the most efficient use of a possibly limited computation time. A viable heuristic

could be based on the size of the facet and include a check whether an element in the outer polytope is given at a minimum step size from the barycenter of a facet in the direction of the outer unit normal. A prioritization strategy is presented in [27].
A parallelization of the online convex hull approach is conceivable by specifying search directions $d^1, \dots, d^{n_{\text{stop}}}$ in advance. From the found vertices, an initial simplex can be formed, and the inner polytope can be built up successively according to Algorithm F. It has to be considered for the original Algorithm E as well as for a parallelized variation that the exact number of facets of the final inner polytope is not predictable. Therefore, if dynamic memory allocation is not possible, a final number of facets must be estimated for pre-allocation. This number can be chosen based on (2.5). Reaching it while the inner polytope is updated thus also represents a termination criterion.
If the underlying feasibility problem is convex and Algorithm E is performed, for each element $z \in \mathcal{P}_{\text{inner}}$, a feasible $x \in \mathbb{R}^{n_{\text{opt}}}$ subject to (g, h) can be determined as a convex combination of the solutions from E-8.1 based on Proposition 4.9. For this, the equation (4.15) must be solved, which for $k > n_{\mathcal{S}} + 1$ is an under-determined linear system of equations subject to inequalities. Further optimization becomes necessary only if there is a requirement that x must minimize a separate cost function.



(a) After 6 optimizations

(b) After 48 optimizations

Figure 4.13: Approximation of reachable set of the Rayleigh problem using the online convex hull approach

In case $\mathcal{S}$ is not convex, $\mathcal{P}_{\text{inner}}$ is not necessarily an under-approximation. This can be observed in Figure 4.13 in which results of the approximation of the reachable set of the Rayleigh problem is illustrated. The polygon outlined in blue represents $\mathcal{P}^k_{\text{inner}}$ and the one in orange $\mathcal{P}^k_{\text{outer}}$ for $k \in \{6, 48\}$. After six optimizations, it can be seen in Figure 4.13a that $\mathcal{P}^6_{\text{inner}}$ is not a subset of $\mathcal{S}$. After 48 optimizations, the inner polytope and the outer polytope are nearly congruent, and it appears that the convex hull of the non-convex $\mathcal{S}$ is approximated. However, on closer inspection of Figure 4.13b, a region of the reachable set is not covered. At one point of the approximation, a vertex was found that is a local but not a global minimum leading to this incident.

4.4 Defect Hull Approach

The defect hull Algorithm continues the ideas of a parallelized online convex hull approach addressed in Section 4.3 and serves as means to approximate a convex set $\mathcal{S}$ in $n_{\mathcal{S}}$-dimensional space. As in the online convex hull approach, an inner and an outer polytope are created that shall resemble $\mathcal{S}$. The vertices of the inner polytope are also generated through optimizations. The focus in the defect hull approach lies in parallelization and the post-optimization processing of results based on parametric sensitivity analysis. In order to invoke concurrent optimizations, a list of search directions

$$\mathcal{D} := \left\{d^1, \ldots, d^{n_{\mathcal{D}}}\right\} \tag{4.18}$$

is defined at the beginning. The elements in $\mathcal{D}$ are considered to be uniformly distributed points[1] on the unit sphere $\mathbb{S}^{n_{\mathcal{S}}-1} := \left\{y \in \mathbb{R}^{n_{\mathcal{S}}} : \|y\|_2 = 1\right\}$. Generating the convex hull of $\mathcal{D}$ results in a boundary description $(\mathcal{D}, \mathcal{I})$. Thus, any element of $\mathcal{I}$ corresponds to a facet of $\operatorname{conv}(\mathcal{D})$:

$$\mathcal{F} \text{ is a facet of } \operatorname{conv}(\mathcal{D}) \iff \exists! I \in \mathcal{I} \colon \mathcal{F} = \operatorname{conv}\left(d^{I_1}, \ldots, d^{I_{n_{\mathcal{S}}}}\right). \tag{4.19}$$

The optimizations required in the defect hull approach are defined as follows.

Definition 4.11 (Defect hull optimization task). *Let (g, h) denote a feasibility problem. Given a vantage point $o \in \mathring{\mathcal{S}}$ and a direction $d \in \mathbb{R}^{n_{\mathcal{S}}}$, the minimization task*

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_{opt}}, \eta \in \mathbb{R}} \quad & -\eta \\ \textit{subject to} \quad & g(x) = 0 \\ & h(x) \leq 0 \\ & z = o + \eta d \end{aligned} \tag{DH(o,d)}$$

with $z = Px$ as in (4.1)*, is called a defect hull task.*

Provided a vantage point $o \in \mathring{\mathcal{S}}$ and a direction $d \in \mathcal{D}$, a minimizer x^* of DH(o,d) is sought such that z^* represents the furthest point on the ray that starts at o and extends along d. This is achieved by additional $n_{\mathcal{S}}$ linear constraints and one more optimization variable η, which represents the distance between z and o. By this means, DH$(o,\cdot)\colon \mathbb{S}^{n_{\mathcal{S}}-1} \to \partial\mathcal{S}$, $d \mapsto z$ is injective. In case the underlying feasibility problem (g,h) is convex, DH(o,d) is a CP.
Furthermore, in the defect hull optimization task, artificial perturbation parameters are incorporated. After finding nominal solutions, they lead to further very valuable properties resulting from the parametric sensitivity analysis.

[1] In this work, uniformly distributed points on the unit sphere were generated using the `Matlab` software package `MinimumEnergyPoints` based on the results in [45].

Definition 4.12 (Perturbed defect hull optimization task). *Let (g,h) denote a feasibility problem. Given a vantage point $o \in \mathring{\mathcal{S}}$ and a direction $d \in \mathbb{R}^{n_\mathcal{S}}$, the minimization task*

$$\begin{aligned} \min_{x\in\mathbb{R}^{n_{opt}},\eta\in\mathbb{R}} \quad & -\eta \\ \textit{subject to} \quad & g(x) = 0 \\ & h(x) \leq 0 \\ & z = o + \eta(d+p) + q \end{aligned} \tag{DH$(o+q, d+p)$}$$

with $z = Px$ as in (4.1), is the perturbed version of DH(o,d). The nominal values for the perturbations are defined as $q_0 := 0$ and $p_0 := 0$.

Both perturbations q and p affect the additional linear constraints introduced in the defect hull task. They are perturbed constantly by q and linearly by p.
The defect hull approach is described in Algorithm G and visualized in Figure 4.14. As inputs in G-1, a feasibility problem (g,h), search directions $\mathcal{D}$, ordered identifiers $\mathcal{I}$, and an interior point o of $\mathcal{S}$ are assumed. Since the search directions are known in advance, they are independent of each other during the optimization step. Thus, all defect hull tasks can be computed concurrently in G-2. Unlike the online convex hull approach, no convex hull is evaluated here. Instead we regard a star-shaped triangulated polytope $\mathcal{P}_{\text{inner}}$ with vertices $\text{DH}(o,\mathcal{D}) := \left\{z^{*,1},\ldots,z^{*,n_\mathcal{D}}\right\}$. The set of offline determined ordered identifiers $\mathcal{I}$ that fulfills (4.19) will be used for the boundary description $(\text{DH}(o,\mathcal{D}),\mathcal{I})$ of the under-approximating inner polytope $\mathcal{P}_{\text{inner}}$. In Figure 4.14, the search directions in $\mathcal{D}$ are depicted by black arrows, and the $\mathcal{P}_{\text{inner}}$ is outlined in blue.

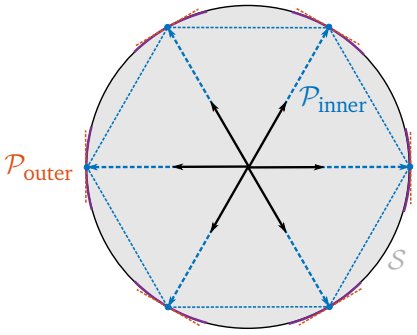


Figure 4.14: Illustration of the defect hull algorithm

It is tolerated that $\mathcal{P}_{\text{inner}}$ in G-3.2 is not convex but star-shaped[2]. The inner polytope is the union of full-dimensional simplices. For each ordered identifier $I \in \mathcal{I}$, such a simplex is given as the convex hull of the found boundary points $z^{*,I_1},\ldots,z^{*,I_{n_\mathcal{S}}}$ and the vantage point o. The simplex partitioning comes along with benefits, which are discussed further

[2]The name of this approach is traced back to the fact that instead of the actual convex hull, a kind of envelope with "dents" is used as an inner approximation. In two-dimensional space, non-convexities do not occur and $\mathcal{P}_{\text{inner}}$ is convex. In three-dimensional space, however, it is different, as the results in Section 4.4.2 show.

in Section 4.4.1.1. The sensitivity-based outer polytope $\mathcal{P}_{\text{outer}}$ is an over-approximation of the convex $\mathcal{S}$. This will be shown in Section 4.4.1.2. In Figure 4.14, the facets of $\mathcal{P}_{\text{outer}}$ are indicated in orange. Another sensitivity result is the curvature hinted in violet.

Algorithm G Defect Hull

G-1: *(Input)*
- G-1.1: *(Feasibility problem)* Provide feasibility problem (g, h)
- G-1.2: *(Search directions)* Provide a set $\mathcal{D} = \{d^1, \dots, d^{n_\mathcal{D}}\} \subset \mathbb{S}^{n_\mathcal{S}-1}$ of uniformly distributed search directions.
- G-1.3: *(Facet identifiers)* Provide a set $\mathcal{I}$ of facet identifiers for which (4.19) holds.
- G-1.4: *(Vantage point)* Provide an interior point $o \in \mathring{\mathcal{S}}$.

G-2: *(Optimization)* Compute $\mathrm{DH}(o, d^i)$ for all d^i, $i = 1, \dots, n_\mathcal{D}$.

G-3: *(Output)*
- G-3.1: *(Results of optimizations)* From defect hull tasks $\mathrm{DH}(o, d^i)$, $i = 1, \dots, n_\mathcal{D}$
  - Primal variables $(x^{*,i}, \eta^{*,i})$ and boundary points $z^{*,i}$
  - Parametric sensitivities $\nabla_q \eta^i(0)$, $\nabla_p \eta^i(0)$, $\nabla_q^2 \eta^i(0)$, $\nabla_p^2 \eta^i(0)$
- G-3.2: *(Inner polytope)* $\mathcal{P}_{\text{inner}} := \bigcup_{I \in \mathcal{I}} \mathrm{conv}(\{o, z^{*,I_1}, \dots, z^{*,I_{n_\mathcal{D}}}\})$
- G-3.3: *(Outer polytope)* $\mathcal{P}_{\text{outer}} := \{y \in \mathbb{R}^{n_\mathcal{S}} : Ay \leq b\}$ where

$$A_i = -\nabla_q \eta^i(0)^\top, \quad b_i = A_i z^{*,i} \tag{4.20}$$

are the respective i-th row of matrix $A \in \mathbb{R}^{n_\mathcal{D} \times n_\mathcal{S}}$ vector $b \in \mathbb{R}^{n_\mathcal{D}}$.

4.4.1 Output Properties

The outputs of Algorithm G provide a variety of properties that can be profitably used after the application. This section explores these properties and is divided into three subsections: First, the inner polytope G-3.2 is examined in more detail. The relationship between the parametric sensitivities and the outer polytope in G-3.3 is explained next. It is possible to access second-order sensitivity derivatives, which could allow sophisticated estimates of the boundary of the sought set to mitigate the need for further optimization.

4.4.1.1 Simplicial Partitions of Inner Polytope

The resulting inner polytope in G-3.2 is the union of simplices whose intersection has no volume. From each of these simplices, the respective volume can be calculated as follows.

Proposition 4.13 (Volume of a partition). *Assume the results of Algorithm G are given provided $\mathcal{D}$, $\mathcal{I}$, and $o \in \mathring{\mathcal{S}}$. Let $I \in \mathcal{I}$ denote an ordered identifier that corresponds to an arbitrary facet of* $\mathrm{conv}(\mathcal{D})$. *Define $\mathcal{P}_I = \mathrm{conv}(\{0, d^{I_1}, \dots, d^{I_{n_\mathcal{S}}}\})$ and $\mathcal{P}'_I = \mathrm{conv}(\{o, z^{*,I_1}, \dots, z^{*,I_{n_\mathcal{S}}}\})$ as partitions of the unit ball polytope and the defect hull. Then,*

$$\mathrm{vol}(\mathcal{P}'_I) = \mathrm{vol}(\mathcal{P}_I) \prod_{j=1}^{n_\mathcal{S}} \left| \eta^*_{I_j} \right|,$$

where $\eta^*_{I_j}$ results from $DH(o, d^{I_j})$, with $z^{*,I_j} = o + \eta^*_{I_j} d^{I_j}$.

Proof. Regard the volume formula given in (2.6). Conclude

$$\begin{aligned}
\operatorname{vol}(\mathcal{P}'_I) &= \frac{1}{n_{\mathcal{S}}!}\left|\det\left(\begin{bmatrix} z^{*,I_1}-o & \dots & z^{*,I_{n_{\mathcal{S}}}}-o \end{bmatrix}\right)\right| \\
&= \frac{1}{n_{\mathcal{S}}!}\left|\det\left(\begin{bmatrix} d^{I_1} & \dots & d^{I_{n_{\mathcal{S}}}} \end{bmatrix}\operatorname{diag}\left(\eta^*_{I_1},\dots,\eta^*_{I_{n_{\mathcal{S}}}}\right)\right)\right| \\
&= \frac{1}{n_{\mathcal{S}}!}\left|\det\left(\begin{bmatrix} d^{I_1} & \dots & d^{I_{n_{\mathcal{S}}}} \end{bmatrix}\right)\right|\left|\det\left(\operatorname{diag}\left(\eta^*_{I_1},\dots,\eta^*_{I_{n_{\mathcal{S}}}}\right)\right)\right| \\
&= \operatorname{vol}(\mathcal{P}_I)\prod_{j=1}^{n_{\mathcal{S}}}\left|\eta^*_{I_j}\right|.
\end{aligned}$$

□

The volume of the partition $\mathcal{P}'_I$ can be directly obtained from the volume of $\mathcal{P}_I$. The total volume of $\mathcal{P}_{\text{inner}}$ can be determined with formula (2.7) which is further simplified through Proposition 4.13.
If $\mathcal{S}$ is convex, the inner polytope is an under-approximation of the sought set, i.e. $\mathcal{P}_{\text{inner}} \subseteq \mathcal{S}$. The procedure to check if a given element is in the inner polytope relies on the following proposition.

Proposition 4.14 (Elements in $\mathcal{P}_{\text{inner}}$). *Assume the results of Algorithm G are given provided $\mathcal{D}$, $\mathcal{I}$, and $o \in \mathring{\mathcal{S}}$. Let $y \in \mathbb{R}^{n_{\mathcal{S}}}$ be an arbitrary point. Then, $y \in \mathcal{P}_{inner}$ holds if and only if an $I \in \mathcal{I}$ exists such that*

$$c := \operatorname{diag}\left(\eta^*_{I_1},\dots,\eta^*_{I_{n_{\mathcal{S}}}}\right)^{-1}\begin{bmatrix} d^{I_1} & \dots & d^{I_{n_{\mathcal{S}}}} \end{bmatrix}^{-1}(y-o) \in \mathbb{R}^{n_{\mathcal{S}}}, \tag{4.21}$$

where $\eta^*_{I_j}$ results from $DH\left(o, d^{I_j}\right)$, with $z^{*,I_j} = o + \eta^*_{I_j} d^{I_j}$, fulfills the following conditions:

$$\begin{aligned}
&c_i \geq 0,\ i = 1,\dots,n_{\mathcal{S}} \\
&\sum_{i=1}^{n_{\mathcal{S}}} c_i \leq 1.
\end{aligned} \tag{4.22}$$

Proof. It follows from $y \in \mathcal{P}_{\text{inner}}$ that a simplex partition $\operatorname{conv}\left(\left\{o, z^{*,I_1},\dots,z^{*,I_{n_{\mathcal{S}}}}\right\}\right)$ for an $I \in \mathcal{I}$ must exist that y is an element of. Thus, $c_0,\dots,c_{n_{\mathcal{S}}} \geq 0$ exist with $\sum_{i=0}^{n_{\mathcal{S}}} c_i = 1$ such that y is a convex combination:

$$c_0 o + \sum_{i=1}^{n_{\mathcal{S}}} c_i z^{*,I_i} = y$$

$$c_0 o + \sum_{i=1}^{n_S} c_i \left(\eta_{I_i} d^{I_i} + o\right) = y$$

$$\begin{bmatrix} d^{I_1} & \dots & d^{I_{n_S}} \end{bmatrix} \operatorname{diag}\left(\eta_{I_1}, \dots, \eta_{I_{n_S}}\right) \begin{pmatrix} c_1 \\ \vdots \\ c_{n_S} \end{pmatrix} = y - o. \tag{4.23}$$

From (4.23), obtain (4.21). The inverse of the matrices on the left hand side of (4.23) are well-defined because $d^{I_1}, \dots, d^{I_{n_S}}$ are linearly independent, and $\eta_{I_j} > 0$ holds for all $j = 1, \dots, n_S$. Finally, (4.22) holds because

$$\sum_{i=1}^{n_S} c_i = 1 - c_0 \leq 1.$$

The reverse implication holds by defining

$$c_0 := 1 - \sum_{i=1}^{n_S} c_i.$$

Then, $c_0, c_1, \dots, c_{n_S}$ are given such that y is a convex combination of $o, z^{*,I_1}, \dots, z^{*,I_{n_S}}$ and therefore an element of $\mathcal{P}_{\text{inner}}$. □

Thus, to check if $y \in \mathcal{P}_{\text{inner}}$ holds, an $I \in \mathcal{I}$ must exist that fulfills (4.21) and (4.22). Furthermore, if the feasibility problem (g, h) is convex, the obtained coefficients (4.21) can be used to assign a feasible $x \in \mathbb{R}^{n_{\text{opt}}}$ to y.

Corollary 4.15 (Pinpoint a feasible solution). *Assume the defect hull algorithm is performed for a convex feasibility problem (g, h), and suppose an $x^0 \in \mathcal{X}$ is given such that $Px^0 = o$, analogously to* (4.1). *Let $z \in \mathcal{P}_{inner}$ be an element of the partition* $\operatorname{conv}\left(\left\{o, z^{*,I_1}, \dots, z^{*,I_{n_S}}\right\}\right)$ *defined by facet identifier $I \in \mathcal{I}$. Furthermore, let $c \in \mathbb{R}^{n_S}$ be defined as in* (4.21) *and fulfill* (4.22). *Then,*

$$x = \left(1 - \sum_{i=1}^{n_S} c_i\right) x^0 + \begin{bmatrix} x^{*,I_1} & \dots & x^{*,I_{n_S}} \end{bmatrix} c \tag{4.24}$$

is feasible subject to (g, h), and $z = Px$.

Proof. The convex feasible set $\mathcal{X}$ contains all convex combinations of its elements, especially (4.24). Multiplying with projection matrix P as in (4.1) on both sides of (4.24) implies $z = Px$. □

In conclusion, the simplex partitioning of the inner polytope can be exploited for volume calculations and a (parallelized) check whether a given point is inside the approximation. If that is the case and in addition, the underlying feasibility problem is convex, a feasible solution can be instantly specified leading to this point.

4.4.1.2 Over-Approximation

If the sought set is convex, the resulting outer polytope is an over-approximation. The first-order sensitivities in G-3.1 represent supporting hyperplanes. The following lemma is necessary to show that.

Lemma 4.16 (Linearity for perturbations in search direction). *Let $z^* \in \mathbb{R}^{n_S}$ denote a boundary point of a convex $\mathcal{S}$ that results as the nominal solution of the optimization problem DH$(o+q,d)$ for $q = q_0$. For DH$(o+q,d)$, the assumptions in the Theorem 3.40 shall be fulfilled. Then, the following holds:*

i. $\nabla_q \eta(0)^\top d = -1$

ii. $d^\top \nabla_q^2 \eta(0) d = 0.$

Proof. For a perturbation $q := \varepsilon d$ along the search direction,

$$z(0) = z(\varepsilon d)$$

holds because DH$(o+q,d)$ would lead to the same boundary point as in the nominal case. This is equivalent to

$$o + \eta(0)\, d = o + \eta(\varepsilon d)\, d + \varepsilon d,$$

which leads to a relation

$$\eta(\varepsilon d) = \eta(0) - \varepsilon.$$

Defining a function $\tilde{\eta}\colon \mathbb{R} \to \mathbb{R}, \varepsilon \mapsto \eta(\varepsilon d)$, $\tilde{\eta}'(0) = -1$ and $\tilde{\eta}''(0) = 0$ are concluded. Applying the chain rule, it follows $\tilde{\eta}'(0) = \nabla_q \eta(0)^\top d$ and $\tilde{\eta}''(0) = d^\top \nabla_q^2 \eta(0) d$, and thus, the statement holds true. □

Theorem 4.17 (q-Sensitivity Induced Supporting Hyperplane). *Let $z^* \in \mathbb{R}^{n_S}$ denote a boundary point of a convex $\mathcal{S}$ which is obtained through the nominal solution of the optimization problem DH$(o+q,d)$ for $q = q_0$. For DH$(o+q,d)$, the assumptions in Theorem 3.40 shall be fulfilled. Then, $\mathcal{S}$ is fully contained in the halfspace induced by the supporting hyperplane*

$$\mathcal{H} := \{x \in \mathbb{R}^n \colon a^\top x = a^\top z^*\} \quad \textit{with } a = \pm \nabla_q \eta(0).$$

Proof. According to the sensitivity theorem a neighborhood $\mathcal{N}_q(0)$ exists such that the differentiable function $z\colon \mathcal{N}_q(0) \to \mathbb{R}^{n_S}$, $q \mapsto z(q)$ maps a perturbation q to a boundary point $z(q) \in \partial\mathcal{S}$ found through DH$(o+q,d)$. Furthermore, $z(0) = z^*$ holds. Since $\mathcal{S}$ is convex, a supporting hyperplane exists at the boundary point $z(0)$. As in the statement, this hyperplane shall be induced by a vector $a \in \mathbb{R}^{n_S} \setminus \{0\}$. Define function $\phi : \mathring{\mathcal{N}}_q(0) \to \mathbb{R}$, $q \mapsto a^\top z(q)$, and note that ϕ is differentiable. For all $\tilde{z} \in \mathcal{S}$,

$$a^\top \tilde{z} \le a^\top z(0)$$

holds, and especially, for arbitrary $q \in \mathcal{N}_q(0)$,

$$a^\top z(q) \le a^\top z(0) \quad \Longleftrightarrow \quad \phi(q) \le \phi(0)$$

applies. Thus, 0 is the maximizer of ϕ, and hence, $\nabla_q \phi(0) = 0$. Recall $z(q) = o + \eta(q)\, d + q$ from DH$(o+q, d)$. From the vanishing gradient

$$\begin{aligned} 0 = \nabla_q \phi(0) &= \nabla_q \left(a^\top z(q)\right)\Big|_{q=0} = \nabla_q a^\top \left(o + \eta(q)\, d + q\right)\Big|_{q=0} \\ &= a^\top d \nabla_q \eta(0) + a, \end{aligned}$$

obtain

$$a = -a^\top d \nabla_q \eta(0). \tag{4.25}$$

Thus, a is a scaled version of $\nabla_q \eta(0)$. Its direction is decided by the second derivative, i.e. the Hessian of ϕ at $q = 0$, which must be negative semi-definite

$$\nabla_q^2 \phi(0) = a^\top d \nabla_q^2 \eta(0) \leq 0. \tag{4.26}$$

Choosing $a = \pm \nabla_q \eta(0)$ fulfills equation (4.25) because $\pm \nabla_q \eta(0)^\top d = \mp 1$ holds due to Lemma 4.16. □

Since $\mathcal{S}$ is assumed to be convex, the enclosed angle between d and an outer normal a of a supporting hyperplane must be acute. This suggests that the inner product $a^\top d$ must be positive. The lower halfspace $\mathcal{H}_\leq$ that contains $\mathcal{S}$ is therefore defined through $a = -\nabla_q \eta(0)$. Alternatively, to avoid checking the Hessian, the direction of a must conform to the inequality $a^\top o \leq a^\top z^*$. The outer polytope $\mathcal{P}_{\text{outer}}$ is the intersection of halfspaces induced by the supporting hyperplanes of all optimizations performed and, thus, an over-approximation of $\mathcal{S}$.
The gradient $\nabla_p \eta$, which has not been investigated yet, is related to $\nabla_q \eta$ in the following way.

Proposition 4.18 (First-order sensitivity conversion). *Let η^* denote the optimal step size found as nominal solution of DH$(o+q, d+p)$ for $q = q_0$ and $p = p_0$. For DH$(o+q, d+p)$, the assumptions in Theorem 3.40 shall be fulfilled. For the first-order sensitivity derivatives $\nabla_p \eta(0)$ and $\nabla_q \eta(0)$ for DH$(o+q, d+p)$, the following relation holds*

$$\nabla_p \eta(0) = -\eta^* \nabla_q \eta(0). \tag{4.27}$$

Proof. Let $\bar{\lambda} \in \mathbb{R}^{n_S}$ denote the Lagrange multiplier that is associated with the constraints perturbed by q and p. Then, according to Corollary 3.41

$$\nabla_q \eta = \bar{\lambda} \tag{4.28}$$

holds. Following Theorem 3.40,

$$\nabla_p \eta = -\nabla_p L = -\eta^* \bar{\lambda} \tag{4.29}$$

can be obtained. Inserting (4.28) into (4.29) leads to the statement. □

Thus, $\nabla_p \eta$ can also theoretically be used to define the outer polytope $\mathcal{P}_{\text{inner}}$.

Corollary 4.19 (p-Sensitivity induced supporting hyperplane). *Let $z^* \in \mathbb{R}^n$ denote the sample point on the boundary of a convex set $\mathcal{S}$ which result as the nominal solution of the optimization problem DH$(o+q, d+p)$ for $q = q_0$ and $p = p_0$. For DH$(o+q, d+p)$, the conditions in Theorem 3.40 are assumed to be fulfilled. Then, $\mathcal{S}$ is fully contained in the halfspace induced by the supporting hyperplane*

$$\mathcal{H} := \{x \in \mathbb{R}^n : a^\top x = a^\top z^*\} \quad \text{with } a = \mp \nabla_p \eta(0).$$

Proof. In the proof of Theorem 4.17, any normal vector $a = \alpha \nabla_q \eta$ with $\alpha \in \mathbb{R} \setminus \{0\}$ is suitable to define the supporting hyperplane, especially $\alpha = -\eta^*$. □

4.4.1.3 Post-Optimal Second-Order Approximation

In Definition 4.12, the perturbation p in DH$(o+q, d+p)$ is introduced as an alternate approach to harness more knowledge from one optimization. Generally, in the defect hull approach, the vantage point o remains the same, while the search direction d is adjusted during the optimizations in G-2. Thus, from an algorithmic point of view, it is more attractive to experience how a solution of DH(o, d) will change when the search direction varies as in DH$(o, d+p)$. The sensitivities resulting from an additive perturbation on the search direction allows well-founded estimations of boundary points surrounding the nominal sample point. The second-order sensitivitites subject to q-type perturbation can be converted.

Proposition 4.20 (Second-order sensitivity conversion). *Let η^* denote the optimal step size found as nominal solution of DH$(o+q, d+p)$ for $q = q_0$ and $p = p_0$. For DH$(o+q, d+p)$, the assumptions in Theorem 3.40 shall be fulfilled. For the second-order sensitivity derivatives $\nabla_p^2 \eta(0)$ and $\nabla_q^2 \eta(0)$ for DH$(o+q, d+p)$, the following relation holds*

$$\nabla_p^2 \eta(0) = 2\eta^* \nabla_q \eta(0) \nabla_q \eta(0)^\top + (\eta^*)^2 \nabla_q^2 \eta(0). \tag{4.30}$$

Proof. For the (i, j)-component of $\nabla_p^2 \eta$, after resolving the dyadic product in (4.30), it must hold:

$$\frac{\partial^2}{\partial p_j \partial p_i} \eta = 2\eta^* \frac{\partial}{\partial q_i} \eta \frac{\partial}{\partial q_j} \eta + (\eta^*)^2 \frac{\partial^2}{\partial q_j \partial q_i} \eta. \tag{4.31}$$

Due to Proposition 4.18, we may start our derivation with

$$\frac{\partial^2}{\partial p_j \partial p_i} \eta = -\frac{\partial}{\partial p_j} \eta \bar{\lambda}_i$$

continue by using the product rule

$$\begin{aligned} &= -\left(\left(\frac{\partial}{\partial p_j} \eta\right) \bar{\lambda}_i + \eta^* \left(\frac{\partial}{\partial p_j} \bar{\lambda}_i\right)\right) \\ &= -\left(-\eta^* \bar{\lambda}_j \bar{\lambda}_i + \eta^* \frac{\partial}{\partial p_j} \left(\frac{\partial}{\partial q_i} \eta\right)\right) \end{aligned}$$

and due to the symmetry of second derivatives (cf. Theorem 1 on page 50 in [32])

$$= -\left(-\eta^* \bar{\lambda}_j \bar{\lambda}_i + \eta^* \frac{\partial}{\partial q_i} \left(\frac{\partial}{\partial p_j} \eta\right)\right)$$

$$
\begin{aligned}
&= -\left(-\eta^* \bar{\lambda}_j \bar{\lambda}_i + \eta^* \left(-\tfrac{\partial}{\partial q_i} \eta \bar{\lambda}_j\right)\right) \\
&= -\left(-\eta^* \bar{\lambda}_i \bar{\lambda}_j - \eta^* \left(\left(\tfrac{\partial}{\partial q_i} \eta\right) \bar{\lambda}_j + \eta^* \left(\tfrac{\partial}{\partial q_i} \bar{\lambda}_j\right)\right)\right) \\
&= \eta^* \bar{\lambda}_j \bar{\lambda}_i + \eta^* \left(\bar{\lambda}_i \bar{\lambda}_j + \eta^* \left(\tfrac{\partial^2}{\partial q_i \partial q_j} \eta\right)\right) \\
&= 2\eta^* \bar{\lambda}_i \bar{\lambda}_j + (\eta^*)^2 \tfrac{\partial}{\partial q_i} \bar{\lambda}_j,
\end{aligned}
$$

which is equivalent to (4.31). □

Assume that the nominal results $z^{*,i}, \eta_i^*, \nabla\eta^i, \nabla_p^2 \eta^i$ from the defect hull optimization task $\mathrm{DH}\big(o, d^i + p\big)$ are given for any $d^i \in \mathcal{D}$ and $p = p_0$. With the available information about the second-order derivative and based on the constraint

$$z = o + \eta\big(d^i + p\big),$$

it is now possible to calculate a Taylor polynomial of corresponding order for the function z^i in $p_0 = 0$. This leads to an approximation

$$
\begin{aligned}
& z^i(p) \approx \tilde{z}^i(p) = \left(\tilde{z}_1^i(p), \ldots, \tilde{z}_{n_\mathcal{S}}^i(p)\right)^\top \\
\text{with} \quad & \tilde{z}_j^i(p) = z_j^{*,i} + \left(d_j^i \nabla_p \eta^i(0) + \eta_i^* e^j\right) p \\
& \qquad + \tfrac{1}{2} p^\top \left(d_j^i \nabla_p^2 \eta^i(0) + e^j \nabla_p \eta^i(0)^\top + \nabla_p \eta^i(0) \left(e^j\right)^\top\right) p
\end{aligned}
\tag{4.32}
$$

where e^j denotes the j-th standard unit vector, $j = 1, \ldots, n_\mathcal{S}$. An alternate approach is to calculate the Taylor polynomial for η_i. Accordingly, obtain

$$
\begin{aligned}
& z^i(p) \approx \tilde{z}^i(p) = o + \tilde{\eta}_i(p)\big(d^i + p\big) \\
\text{with} \quad & \tilde{\eta}_i(p) = \eta_i^* + \nabla_p \eta^i(0)^\top p + \frac{1}{2} p^\top \nabla_p^2 \eta^i(0)\, p.
\end{aligned}
\tag{4.33}
$$

After applying the defect hull algorithm, an approximation based on the results of all optimization tasks $\mathrm{DH}\big(o, d^i\big)$, $d^i \in \mathcal{D}$, is applicable. For this purpose a function

$$s \colon \mathbb{S}^{n_\mathcal{S}-1} \to \mathbb{R}^{n_\mathcal{S}} \tag{4.34}$$

is established. It maps a point d of the unit sphere to the approximate boundary point of $\mathcal{S}$ based on solutions of $\mathrm{DH}\big(o, d^i + p\big)$ for d^i nearest to d:

$$
s(d) = \begin{cases}
\tilde{z}^1\left(d - d^1\right), & \text{if } d^1 = \arg\min\left\{-d^\top d^i \colon d^i \in \mathcal{D}\right\} \\
\vdots & \vdots \\
\tilde{z}^{n_\mathcal{D}}\left(d - d^{n_\mathcal{D}}\right), & \text{if } d^{n_\mathcal{D}} = \arg\min\left\{-d^\top d^i \colon d^i \in \mathcal{D}\right\}
\end{cases}
\tag{4.35}
$$

with either definition (4.32) or (4.33). By function s, an approximate boundary reconstruction of the sought set $\mathcal{S}$ is given. The underlying defining search directions in $\mathcal{D}$ can

be supplemented by further ones. With the circumstances of the defect hull algorithm, the search directions

$$d^I := \frac{\sum_{i=1}^{n_\mathcal{S}} d^{I_i}}{\left\| \sum_{i=1}^{n_\mathcal{S}} d^{I_i} \right\|_2}, \; I \in \mathcal{I} \tag{4.36}$$

are obvious candidates. One way to prioritize or select them is based on the additional volume that can be covered in case of an optimization. For an ordered identifier $I \in \mathcal{I}$, the search direction d^I leads to an approximate boundary point $z^I := s\left(d^I\right)$. Due to (2.6), the additional volume is given by

$$\operatorname{vol}\left(\operatorname{conv}\left(\left\{z^I, z^{*,I_1}, \ldots, z^{*,I_{n_\mathcal{S}}}\right\}\right)\right) = \frac{1}{n_\mathcal{S}!} \left| \det\left(\begin{bmatrix} z^{*,I_1} - z^I & \ldots & z^{*,I_{n_\mathcal{S}}} - z^I \end{bmatrix} \right) \right|$$

which can be reformulated to

$$= \frac{1}{n_\mathcal{S}!} \left| \det\left(\begin{bmatrix} \eta^*_{I_1} d^{I_1} & \ldots & \eta^*_{I_{n_\mathcal{S}}} d^{I_{n_\mathcal{S}}} \end{bmatrix} + \left(o - z^I\right) \begin{pmatrix} 1 & \ldots & 1 \end{pmatrix} \right) \right|$$

hinting the application of the *matrix determinant lemma* (cf. Lemma 1.1 in [23])

$$\begin{aligned} = \frac{1}{n_\mathcal{S}!} \Bigg| & \det\left(\begin{bmatrix} \eta^*_{I_1} d^{I_1} & \ldots & \eta^*_{I_{n_\mathcal{S}}} d^{I_{n_\mathcal{S}}} \end{bmatrix} \right) \\ & \cdot \left(1 + \begin{pmatrix} 1 & \ldots & 1 \end{pmatrix} \begin{bmatrix} \eta^*_{I_1} d^{I_1} & \ldots & \eta^*_{I_{n_\mathcal{S}}} d^{I_{n_\mathcal{S}}} \end{bmatrix}^{-1} \left(o - z^I\right) \right) \Bigg|. \end{aligned}$$

Based on Proposition 4.13 and its definitions, the following relation is deduced:

$$\begin{aligned} & \operatorname{vol}\left(\operatorname{conv}\left(\left\{z^I, z^{*,I_1}, \ldots, z^{*,I_{n_\mathcal{S}}}\right\}\right)\right) \\ & = \operatorname{vol}\left(\mathcal{P}'_I\right) \left| 1 - \begin{pmatrix} \frac{1}{\eta^*_{I_1}} & \cdots & \frac{1}{\eta^*_{I_{n_\mathcal{S}}}} \end{pmatrix} \begin{bmatrix} d^{I_1} & \ldots & d^{I_{n_\mathcal{S}}} \end{bmatrix}^{-1} \left(z^I - o\right) \right|. \end{aligned} \tag{4.37}$$

A dimensionless quantity

$$\rho_I := \frac{\operatorname{vol}\left(\operatorname{conv}\left(\left\{z^I, z^{*,I_1}, \ldots, z^{*,I_{n_\mathcal{S}}}\right\}\right)\right)}{\operatorname{vol}\left(\mathcal{P}'_I\right)}$$

is obtained by dividing by $\operatorname{vol}\left(\mathcal{P}'_I\right)$ on both sides of (4.37). This ends in the following definition.

Definition 4.21 (Partition extension potential). *Let $I \in \mathcal{I}$ be an ordered identifier, and let $\mathcal{P}'_I := \operatorname{conv}\left(\left\{o, z^{*,1}, \dots, z^{*,n_\mathcal{S}}\right\}\right)$ denote a simplex partition of the inner polytope. Furthermore, evaluate $z^I = s\left(d^I\right)$ with d^I as in* (4.36) *for example. The extension potential of the partition with identifier I is given by*

$$\rho_I = \left| 1 - \begin{pmatrix} \frac{1}{\eta^*_{I_1}} & \cdots & \frac{1}{\eta^*_{I_{n_\mathcal{S}}}} \end{pmatrix} \begin{bmatrix} d^{I_1} & \dots & d^{I_{n_\mathcal{S}}} \end{bmatrix}^{-1} \left(z^I - o\right) \right|. \tag{4.38}$$

Thus, the partition extension potential is the fraction of the volume of $\mathcal{P}'_I$ that can be additionally accessed by solving $\mathrm{DH}\left(o, d^I\right)$. The larger ρ_I is, the higher z^I rises above the facet. Accordingly, the curvature of the boundary of the set $\mathcal{S}$ above the facet necessarily changes more. More support points in this area thus improve the quality of the reconstruction with the function s. The direction d_I does not have to be chosen as in (4.36). However, it must be a search direction that can be assigned to the partition I because it is a normalized convex combination of the vertices of the facet. Figure 4.15 shows a two-dimensional example illustrating the partition extension potential. Here, the ordered identifier is given by $I = (i, j)$. The blue partition $\mathcal{P}'_I$ is the simplex with vertices $o, z^{*,i}, z^{*,j}$. In this example, direction d^I chosen as in (4.36). Through the approximation $s\left(d^I\right) = z^I$, another triangle can be constructed with $\operatorname{conv}\left(\left\{z^I, z^{*,i}, z^{*,j}\right\}\right)$ which is colored in pink. The partition extension potential ρ_I is the ratio between the pink and the blue area.

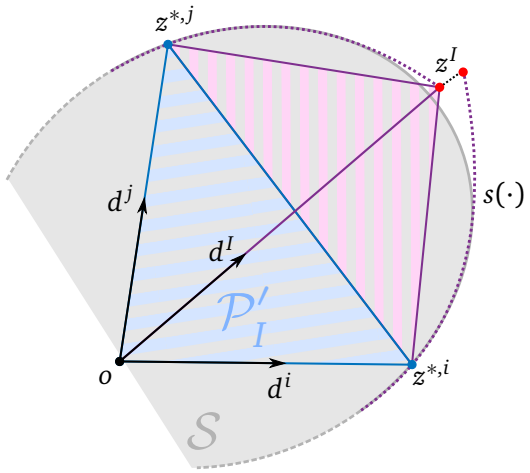


Figure 4.15: Partition extension potential and approximation discrepancy

Because of the construction design of s, there are inevitably evaluation points where at least two operating points are suitable for a Taylor expansion based approximation. This will lead to discontinuities, indicated in Figure 4.15 by two red dots. The *approximation discrepancy* quantifies this discontinuity through the distance between the transition points.

Definition 4.22 (Approximation discrepancy). *Let $I \in \mathcal{I}$ be a ordered identifier of a facet. Regard* (4.32) *or* (4.33) *to evaluate $\tilde{z}^{I_j}\left(d^I - d^{I_j}\right)$ with d^I as in* (4.36). *The approximation*

discrepancy δ_I for d^I is given by

$$\delta_I = \max_{i,j=1,\dots,n_{\mathcal{S}}} \left\| \tilde{z}^{I_i}\left(d^I - d^{I_i}\right) - \tilde{z}^{I_j}\left(d^I - d^{I_j}\right) \right\|_2 \tag{4.39}$$

The definition of the approximation discrepancy can be simplified in case approximation method (4.33) is used.

Corollary 4.23 (Simplified approximation discrepancy). *If in Definition 4.22 approximation method* (4.33) *is used, the approximation discrepancy δ_I simplifies to*

$$\delta_I = \max_{i,j=1,\dots,n_{\mathcal{S}}} \left| \tilde{\eta}_{I_i}\left(d^I - d^{I_i}\right) - \tilde{\eta}_{I_j}\left(d^I - d^{I_j}\right) \right|. \tag{4.40}$$

Proof. This can be shown by inserting (4.33) into (4.39):

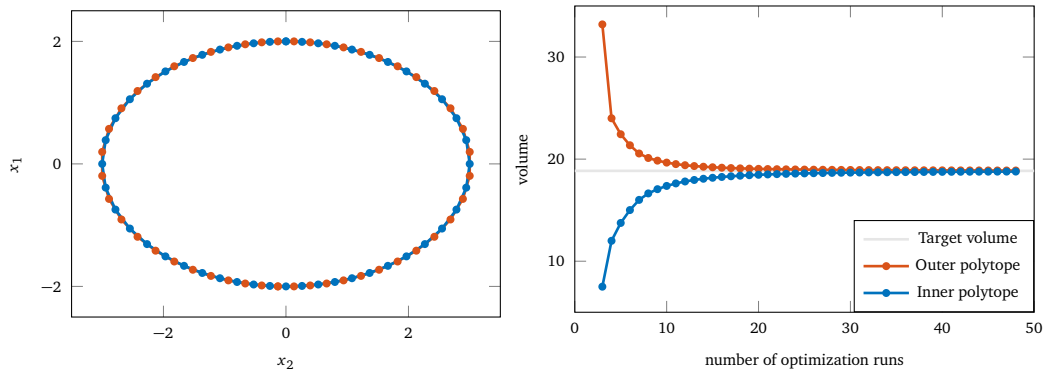
$$\begin{aligned}
\delta_I &= \max_{i,j=1,\dots,n_{\mathcal{S}}} \left\| \tilde{z}^{I_i}\left(d^I - d^{I_i}\right) - \tilde{z}^{I_j}\left(d^I - d^{I_j}\right) \right\|_2 \\
&= \max_{i,j=1,\dots,n_{\mathcal{S}}} \left\| o + \tilde{\eta}_{I_i}\left(d^I - d^{I_i}\right) d^I - \left(o + \tilde{\eta}_{I_j}\left(d^I - d^{I_j}\right) d^I\right) \right\|_2 \\
&= \max_{i,j=1,\dots,n_{\mathcal{S}}} \left\| d^I \right\|_2 \left| \tilde{\eta}_{I_i}\left(d^I - d^{I_i}\right) - \tilde{\eta}_{I_j}\left(d^I - d^{I_j}\right) \right| \\
&= \max_{i,j=1,\dots,n_{\mathcal{S}}} \left| \tilde{\eta}_{I_i}\left(d^I - d^{I_i}\right) - \tilde{\eta}_{I_j}\left(d^I - d^{I_j}\right) \right|
\end{aligned}$$

□

While the partition extension potential could be evaluated for any search direction d^I, the approximation discrepancy are best evaluated for d^I as defined in (4.36).

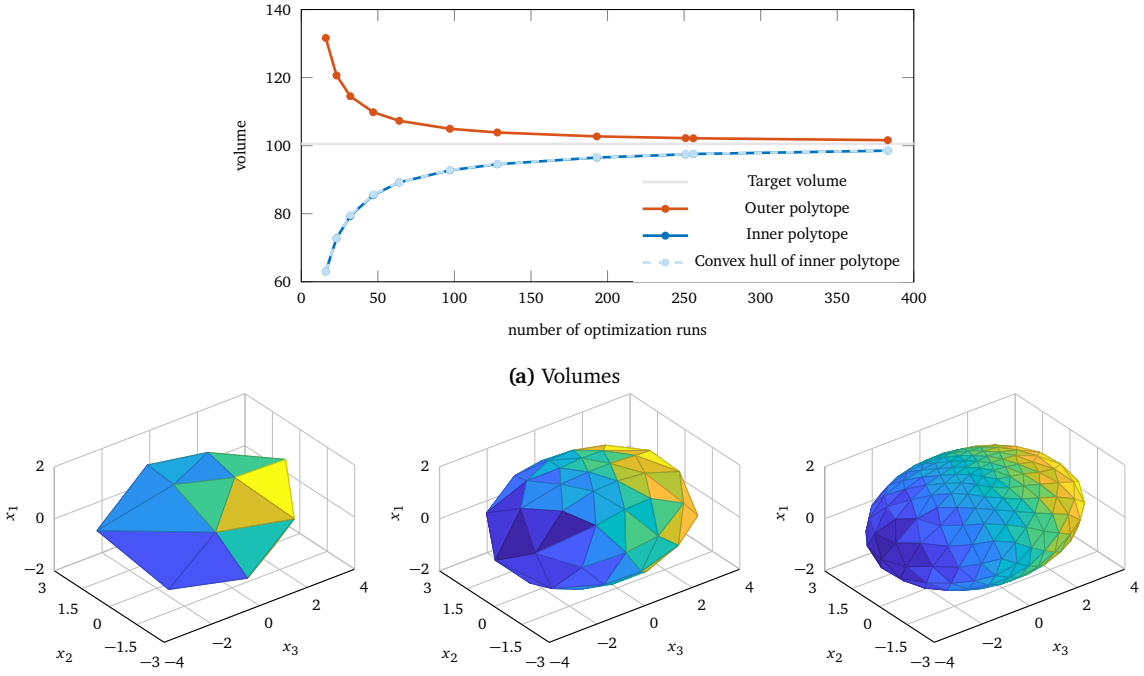
4.4.2 Discussion

The defect hull algorithm produces an approximation of the searched set $\mathcal{S}$ by connecting points found by optimization to form a star-shaped triangulated polytope $\mathcal{P}_{\text{inner}}$ in a predefined manner. This polytope is an under-approximation of a sought $\mathcal{S}$ set that is convex. At the same time, the parametric sensitivities result in another polytope $\mathcal{P}_{\text{outer}}$ in the $\mathcal{H}$-representation, which acts as an over-approximation.
This approach is best suited for approximating convex sets with a smooth boundary. This can be observed in Figure 4.16. The ellipse of feasibility problem (E) with $\alpha_1 = 2, \alpha_2 = 3$ becomes apparent at the latest after 48 optimizations, which is illustrated in Figure 4.16a. The inner polytope outlined in blue and the orange over-approximation are nearly congruent. Algorithm G has been performed for each $n_{\mathcal{D}} \in \{3, \dots, 48\}$ with different sets of search directions to approximate the ellipse. The blue and orange volume curves of the inner and outer polytope exhibit convergence behavior against the target volume in gray in Figure 4.16b. The relative difference between $\mathcal{P}_{\text{inner}}$ and $\mathcal{P}_{\text{outer}}$ is less than 5% after 16 optimizations. It monotonically decreases to about 0.5% applying a set of 48 search directions.



(a) After 48 optimizations **(b)** Volume of inner and outer polytope

Figure 4.16: Approximation of an ellipse using the defect hull approach



(a) Volumes

(b) After 16 optimizations **(c)** After 64 optimizations **(d)** After 256 optimizations

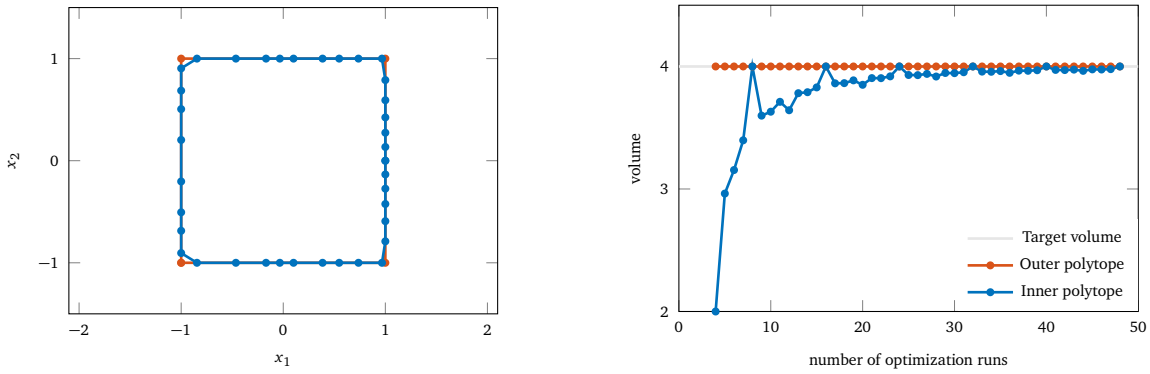
Figure 4.17: Approximation of an ellipsoid using the defect hull approach

For the three-dimensional results, less samples have been computed. Sets of search directions have been prepared for $n_{\mathcal{D}} \in \{16, 23, 32, 47, 64, 97, 128, 193, 251, 256, 383\}$ choosing powers of two and prime numbers. The ellipsoid ((E) with $\alpha_1 = 2$, $\alpha_2 = 3$ and $\alpha_3 = 4$) becomes clearly recognizable in Figure 4.17 the more optimizations are performed. A light blue curve that shows the course of the volume of the convex hull of the inner polytope is added to the convergence plot in Figure 4.17a. The blue and light blue curves are practically the same. The defects are barely visible in the case of the ellipsoid in Figures 4.17b - 4.17d in which the respective outer polytope is omitted for a clearer visualization.

$n_{\mathcal{D}}$	16	23	32	47	64	97	128	193	251	256	383
$\mathrm{vol}(\mathcal{P}_{\mathrm{inner}})$	63.0351	72.7864	79.3533	85.4959	89.2063	92.7621	94.5435	96.5105	97.4482	97.5537	98.5291
$\mathrm{vol}(\mathrm{conv}(\mathcal{P}_{\mathrm{inner}}))$	63.0351	72.7864	79.5698	85.6542	89.2206	92.8153	94.6067	96.5801	97.4823	97.5806	98.5466
Ratio in %	100	100	99.7279	99.8151	99.9839	99.9427	99.9332	99.9280	99.9650	99.9724	99.9823

Table 4.1: Comparison of volumes of the inner approximation of an ellipsoid and its convex hull

This is further underlined by Table 4.1. It shows that the relative difference in volumes, if there is any, is considerably less than 1%. The largest discrepancy in this experiment is given with a set of 32 search directions which leads to 0.27% less coverable volume. On average, the inner approximation comprises 99.93% of its convex hull in this experiment making the defect hull algorithm a considerable option for the approximation of smooth sets.



(a) After 47 optimizations

(b) Volume of inner and outer polytope

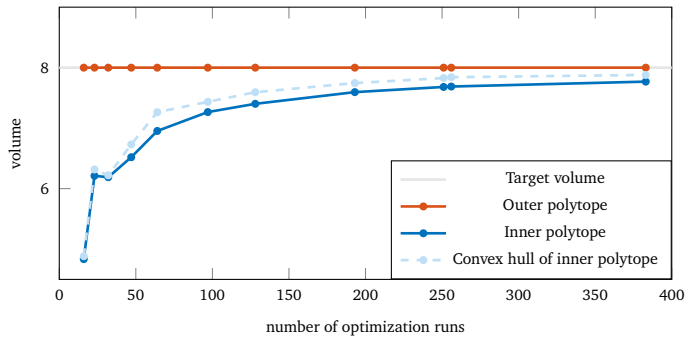
Figure 4.18: Approximation of a square using the defect hull approach

For non-smooth boundaries $\partial\mathcal{S}$, like the square ((B) with $\beta_1 = \beta_2 = 1$) in Figure 4.18, the convergence behavior is hinted but not as regular. For the inner polytope of the approximation to cover the whole square, the four corners must be found during the optimizations. Whether this occurs or not depends on the search directions given as inputs. In this experiment, sets with $n_{\mathcal{D}} \in \{8, 16, 24, 32, 40, 48\}$ search directions achieve a complete reconstruction of the square[3]. The outer polytope reaches the target volume as soon as one boundary point per side excluding the corners have been found. This explains how the outer polytope reaches the target volume from the start[4].
Figure 4.19 illustrates the results for the cube ((B) with $\beta_1 = \beta_2 = \beta_3 = 1$). Despite many performed optimizations, the approximation does not properly takes the form of the cube as characteristics like the corners and edges are not well reconstructed using this approach. The defects due to the saved convex hull evaluations are clearly visible in Figures 4.19c and 4.19d but also in Figure 4.19a, as the blue and light blue curves, representing the volumes of the inner polytope and its convex hull respectively, proceed differently. Nonetheless, the

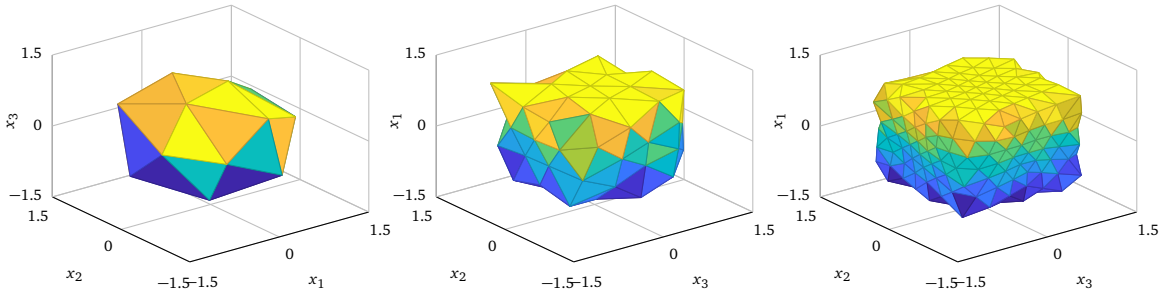
[3]This regularity occurs because the $n_{\mathcal{D}}$ search directions were uniformly distributed over $\mathbb{S}^1$.

[4]Sets of search directions have been prepared for $n_{\mathcal{D}} \in 4, \ldots, 48$ for these results. The case $n_{\mathcal{D}} = 3$ leads to an outer polytope in a $\mathcal{H}$-description where two of the three facets are parallel to each other.

volumes converge. Furthermore, Table 4.2 makes it clear that despite the dents, the inner defect hull approximation comprise a large part of the space the convex hull would. On average, the ratio is 97.95% and in the worst case still more than 95% in this experiment. This underlines the idea that for two- and three-dimensional applications, a predefined boundary description is a viable means of saving computational time.



(a) Volumes



(b) After 16 optimizations **(c)** After 64 optimizations **(d)** After 256 optimizations

Figure 4.19: Approximation of a cube using the defect hull approach

$n_{\mathcal{D}}$	16	23	32	47	64	97	128	193	251	256	383
vol($\mathcal{P}_{\text{inner}}$)	4.8350	6.2119	6.1894	6.5195	6.9535	7.2659	7.4022	7.5954	7.6811	7.6878	7.7690
vol(conv($\mathcal{P}_{\text{inner}}$))	4.8833	6.3175	6.2201	6.7301	7.2638	7.4351	7.5938	7.7453	7.8281	7.8420	7.8794
Ratio in %	99.01	98.33	99.51	96.87	95.73	97.72	97.48	98.06	98.12	98.03	98.60

Table 4.2: Comparison of volumes of the inner approximation of a cube and its convex hull

It is of course possible to form the convex hull of the found vertices in the actual application if resources are available. However, as mentioned in Section 4.3.2, it is not possible to exactly predict for higher dimensions how many elements the boundary description consists of. If the predefined set of ordered identifiers is used, further preparations can be done based on them. The Propositions 4.13 and 4.14 as well as Definition 4.21 have been formulated in a way, that the presented property can be traced back to an operation on matrix

$$D_I := \begin{bmatrix} d^{I_1} & \dots & d^{I_{n_S}} \end{bmatrix} \in \mathbb{R}^{n_S \times n_S}$$

whose columns are search directions in $\mathcal{D}$ specified through $I \in \mathcal{I}$. Though the computation times for the determinant and inversion (or decomposition) of D_I are relatively low, especially for small $n_\mathcal{S}$, a continuous re-evaluation may be avoided provided slightly more memory. Each simplex partition of the inner polytope $\mathcal{P}_{\text{inner}}$ is a subset of $\mathcal{S}$. Therefore, to verify whether a given point $y \in \mathbb{R}^{n_\mathcal{S}}$ is an element of the inner polytope, at least one simplex partition must contain y. For this purpose, according to Proposition 4.14, a $c \in \mathbb{R}^{n_\mathcal{S}}$ is computed whose components are non-negative and in sum must not be greater than 1. Assuming the first condition is fulfilled for a partition with identifier $I \in \mathcal{I}$, $y - o$ is located in the cone defined by the ray directions $d^{I_1}, \dots d^{I_{n_\mathcal{S}}}$. If the sum of the components of c is less than or equal to 1, $y - o$ is in or below the facet that makes the cone a simplex. An example is given through Figure 4.20. Point y^1 is inside the cone that is created through the rays which start at o and extend along d^1 and d^2. Coefficients $c_1, c_2 \geq 0$ of a conic combination can be found such that $y^1 - o = c_1 \left(z^{*,1} - o\right) + c_2 \left(z^{*,2} - o\right)$. Another coefficient $c_0 := 1 - c_1 - c_2 \geq 0$ exists to complete the convex combination $y^1 = c_0 o + c_1 z^{*,1} + c_2 z^{*,2}$, since y^1 is below the line segment that connects $z^{*,1}$ and $z^{*,2}$ and thus in $\mathcal{P}'_{(1,2)}$. If the resulting $c \in \mathbb{R}^{n_\mathcal{S}}$ represents a conic combination but the sum of its components is larger than 1, then the check does not have to be continued because the interiors of the cones do not intersect. In this case, y would not be an element of the inner polytope. This is the case for y^2 in Figure 4.20. The point is located in the same cone as $\mathcal{P}'_{(2,3)}$ therefore it cannot be an element of the interior of $\mathcal{P}'_{(1,2)}$.

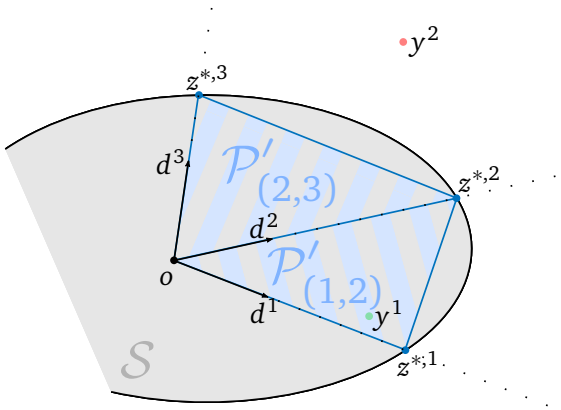
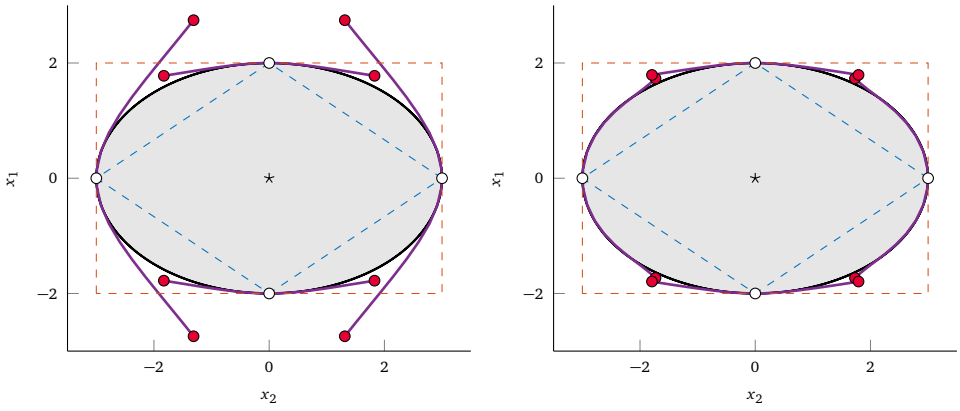


Figure 4.20: Example for application of Proposition 4.14

Furthermore, if the underlying feasibility problem is convex and the resulting $c \in \mathbb{R}^{n_\mathcal{S}}$ represents coefficients of a convex combination, it can be applied to the entire optimization vector as done in (4.24) in Corollary 4.15. This requires an according to (g, h) feasible $x^0 \in \mathbb{R}^{n_{\text{opt}}}$, which satisfies $Px^0 = o$. If x^0 is not yet known, this can be determined by solving the grid task G(o). For a convex feasibility problem (g, h), the results of the defect hull algorithm can therefore be seen as a library to choose a feasible $x \in \mathcal{X}$ as long as $Px \in \mathcal{P}_{\text{inner}}$. The difficulty that exists in Proposition 4.9 that additional inequalities must be considered can be circumvented with a simplex partitioning.

In the defect hull approach, the optimizations in step G-2 consumes the largest part of the computation time. Beside the concurrent solving of the optimization tasks, warm starts should be exploited since the sparsity patterns of internal matrices will not change. However, these warm starts are not necessarily feasible, as the $n_{\mathcal{S}}$ additional constraints in the defect hull task are rarely fulfilled for two distinct search directions at the same time. In case of a failed optimization, a solver instance in another thread should retry the task, as the last solution in the cache represents another initial guess.
From each of the optimizations, sensitivity derivatives can be obtained. Using $\nabla_q \eta$ (or $\nabla_p \eta$), the $\mathcal{H}$-representation of the outer polytope $\mathcal{P}_{\text{outer}}$ can be specified as in G-3.3. Due to Theorem 4.17 (or Corollary 3.41), $\mathcal{P}_{\text{outer}}$ represents an over-approximation of $\mathcal{S}$. Regarding $\text{DH}(o, \cdot)$ as a map sets the foundation for a sensitivity-based reconstruction of $\partial\mathcal{S}$. Through the formulation $\text{DH}(o, d + p)$ derivative information is available for all elements $d \in \mathcal{D}$ in the set of search directions, which, together with the sample points $\text{DH}(o, \mathcal{D}) \subset \mathcal{S}$, can be exploited for interpolation. Each element in $\mathcal{D}$ represents an expansion point of a Taylor polynomial. In (4.35), a nearest-neighbor interpolation approach to estimate $z^* = \text{DH}(o, d)$ for some $d \in \mathbb{S}^{n_S - 1}$ is presented that proposes which expansion point to choose. By this means, function $s \colon \mathbb{S}^{n_S - 1} \to \mathbb{R}^{n_S}, d \mapsto \tilde{z} \approx \text{DH}(o, d)$ is specified.

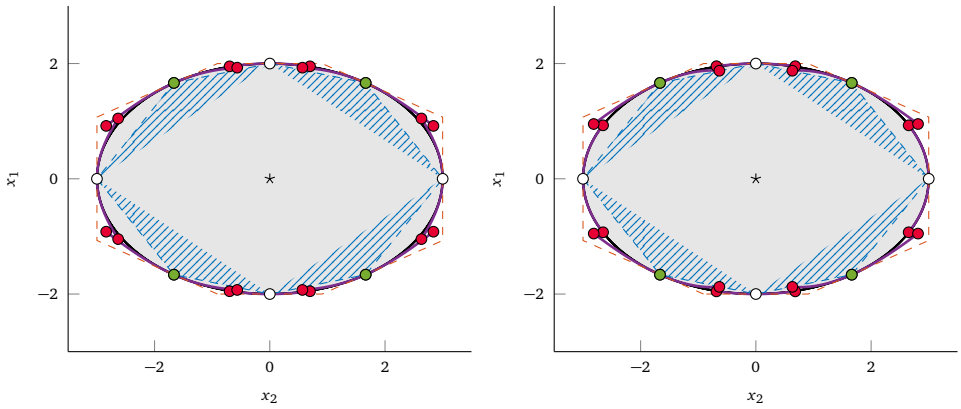


(a) Based on (4.32) **(b)** Based on (4.33)

Figure 4.21: Boundary interpolation of an ellipse

The interpolation is applied on the ellipse defined by (E) with $\alpha_1 = 2$ and $\alpha_2 = 3$ after performing Algorithm G with $n_{\mathcal{D}} = 4$. Figure 4.21 illustrates both approximation types (4.32) and (4.33). The ellipse in gray, $\mathcal{P}_{\text{inner}}$ in blue, $\mathcal{P}_{\text{outer}}$ in orange, and function s in violet are plotted in Figures 4.21a and 4.21b. The white dots represents the points $\text{DH}(o, \mathcal{D})$. With a small number of optimizations, the ellipse can be encapsulated in both cases, although especially towards a non-continuous transition from one expansion point to the next, the error of the estimates increases as expected, as the distance to an expansion point rises. While s leaves $\mathcal{P}_{\text{outer}}$ in Figure 4.21a, the reconstruction remains inside the outer polytope in Figure 4.21b and is significantly closer to $\mathcal{S}$ than the inner and outer approximation.
The refinements at d^I as defined in (4.36) for $I \in \mathcal{I}$ are illustrated in Figure 4.22. They lead to new boundary elements marked in green and extend s by four more expansion

points. The reconstruction based on (4.32) lies within the over-approximation now. The gaps between the transition points are visibly smaller. In Figure 4.22b, an overall reduction of the discontinuities can not be observed. In both cases, the inner polytope grows by the hatched area and function s takes the form of the ellipse in a clearly recognizable way.

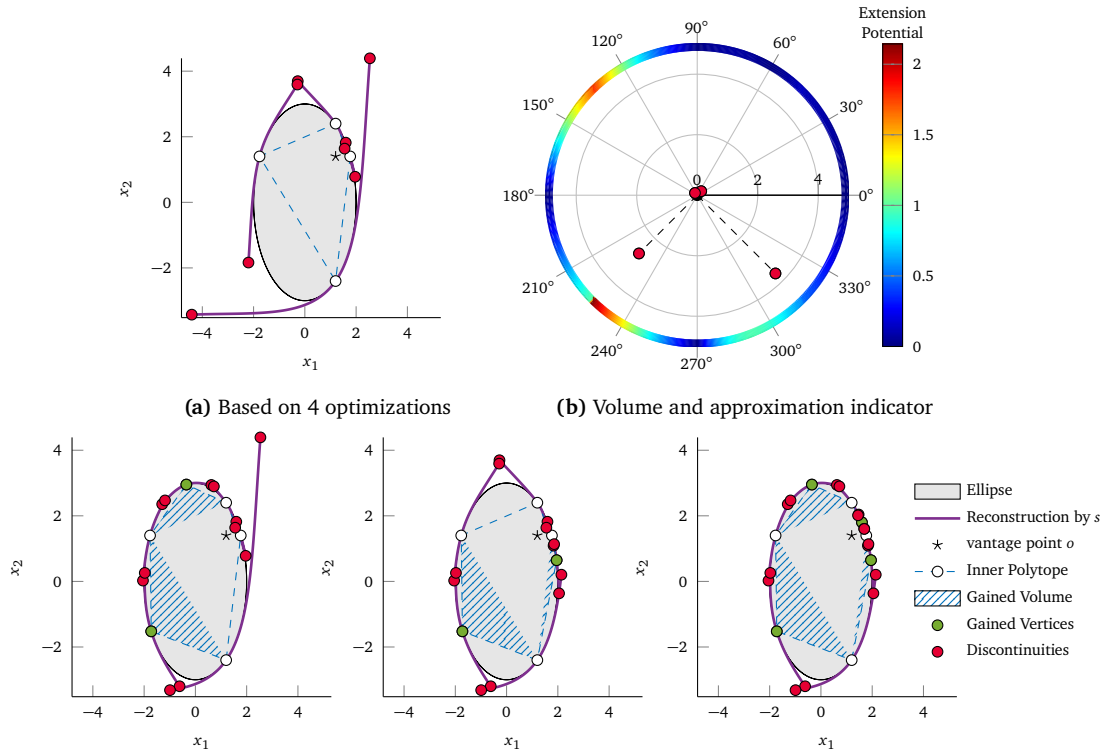


(a) Based on (4.32) **(b)** Based on (4.33)

Figure 4.22: Refined boundary interpolation of an ellipse

Essential for Algorithm G is the vantage point $o \in \mathring{z}$. So far, $o = \begin{pmatrix}0, & 0\end{pmatrix}^\top$ is selected as input for the approximations. If no inner point o results from a user's preliminary analyses, it must be determined online. In the case of a convex set $\mathcal{S}$, the initial simplex of the online convex hull approach can be determined (see E-2 and the discussion in Section 4.3.2), and o can be defined as the center of gravity of this simplex.

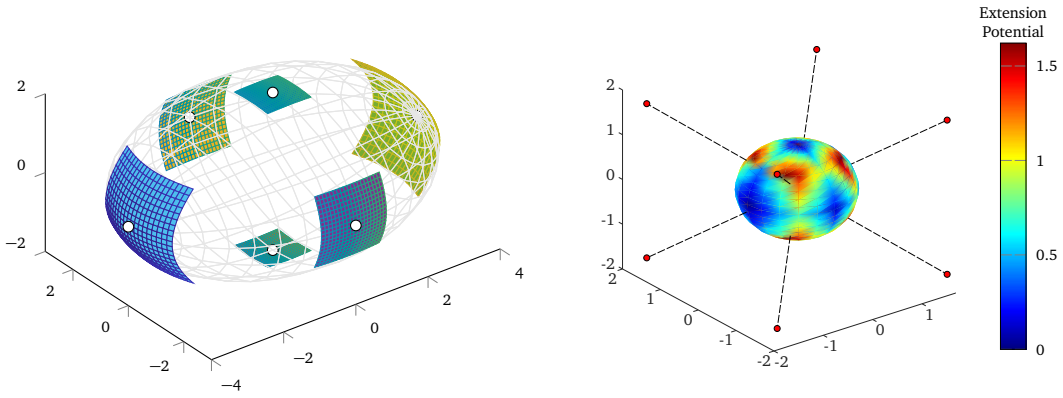
Figure 4.23a shows the approximation of the ellipse, for which $o = \begin{pmatrix}1.2, & 1.4\end{pmatrix}^\top$ is chosen as the vantage point. The reconstruction s is less symmetrical and its gravest discontinuities are located at the lower left (225°) and lower right (315°) of the vantage point. This can be noted from Figure 4.23b, which illustrates the distances of the transition points in a polar plot. Each direction d^I can be converted to polar coordinates with radius 1. The radius is scaled according to the approximation discontinuity and drawn into the polar plot. The circular heatmap is created by evaluating the partition extension potential for all directions $d \in \mathbb{S}^{n_S-1}$. Near already optimized search directions $d \in \mathcal{D}$ the extension potential is about zero. It grows with the distance between s and the inner polytope. The heatmap ring points out in which directions the inner polytope expands more during refinements. Both the approximation discontinuities and the extension potential can be regarded as indicators for prioritization of the refinements. Figure 4.23e serves as a comparison in which $\mathrm{DH}\left(o, d^I\right)$ is solved for all $I \in \mathcal{I}$. Figure 4.23c aims for a high growth of the inner polytope and performs further optimizations in directions given at 135° and 225°. The gained volume through two additional optimizations is in fact maximized for directions defined in (4.36). Figure 4.23d prioritize the removal of discontinuities at 225° and 315°. The updated s adjust well to the ellipse and the distances of transition points is immensely reduced.



(a) Based on 4 optimizations **(b)** Volume and approximation indicator

(c) Volume prioritization **(d)** Approximation prioritization **(e)** Complete Refinement

Figure 4.23: Ellipse refinement

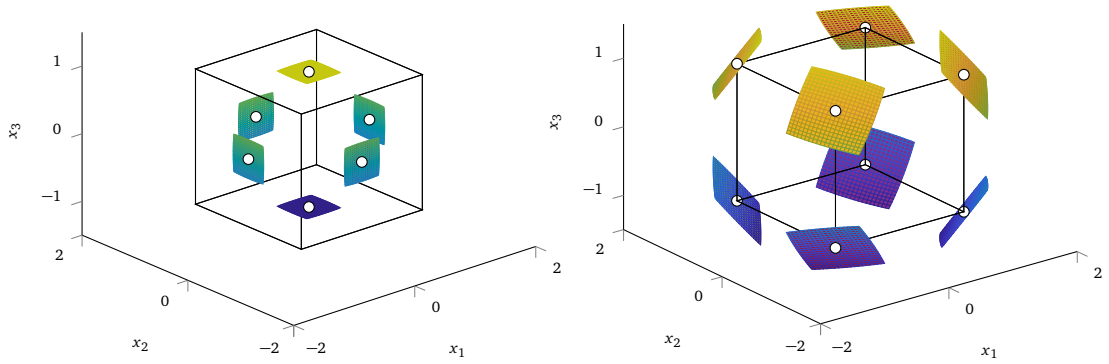


(a) Sensitivity patches on an ellipsoid **(b)** Volume and approximation indicator

Figure 4.24: Reconstruction approach illustrated for three dimensions

The reconstruction approach based on sensitivities and a prioritization of refinements can be transferred to higher dimensions. Figure 4.24a shows patches shaped through slopes and curvatures determined from sensitivities at the respective boundary points marked by white dots. Figure 4.24b is the 3D-version of the polar plot in Figure 4.23b. The exten-

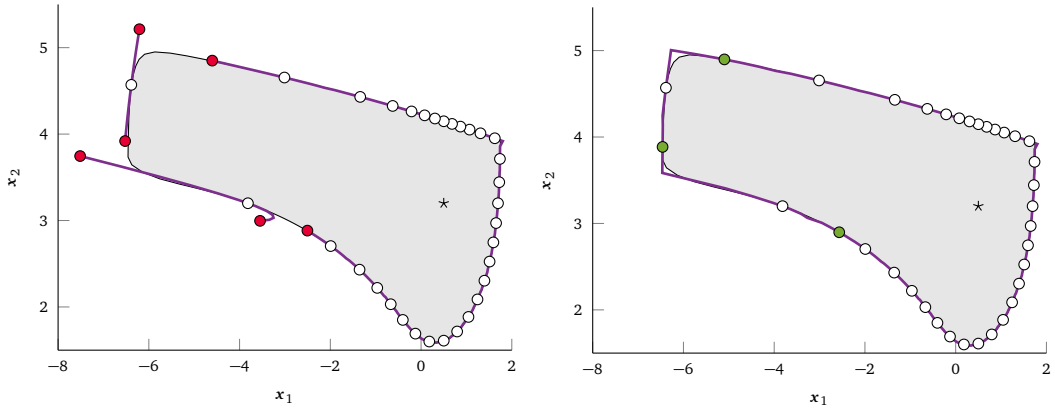
sion potential is given as a heatmap on a sphere, and the approximation discontinuities are given by the distances between the red dots and the origin.



(a) Patches on the sides **(b)** Patches at corners

Figure 4.25: Sensitivities on the surface of a cube

For sets with non-smooth surfaces, the sensitivity based reconstruction with function s will not work. Figure 4.25 is regarded as an example. The cube has zero curvatures at the sides, while edges and corners are not differentiable. Analytically deriving the solutions that leads to the corners of the cube, one comes to the conclusion that the dual variables are not unique at the optimum. Though they exist, the supporting hyperplanes there are not unique either. The ones shown in Figure could be rigorously tilted and would still be feasible. Furthermore, the matrix in (3.29) is not invertible, so no meaningful second-order sensitivities can be expected.



(a) With vantage point at $o = \begin{pmatrix} 0.5, & 3.2 \end{pmatrix}^\top$ **(b)** With additional expansion points

Figure 4.26: Sensitivity-based boundary reconstruction of the Rayleigh problem

An application to star-shaped sets is also possible under certain circumstances. The result of the approximation of the reachable set of the Rayleigh problem is illustrated in Figure 4.26. In the first instance, $n_{\mathcal{D}} = 32$ optimizations are performed leading to the white markers in the plot. The purple sensitivity-based reconstruction adapts to the actual boundary

of the reachable set quite well in parts. However, in Figure 4.26a, this is only limited to the local area around a boundary point resulting from an optimization. Three of the transitions from one Taylor expansion point to the next are especially eye-catching. By evaluating the approximation discrepancy, they can be detected. In Figure 4.26b, three additional expansion points, indicated by green markers, are inserted, and the boundary reconstruction with function s is re-evaluated. The spikes in Figure 4.26a do not appear anymore in Figure 4.26b due to the additional Taylor polynomials.

Chapter 5

Application to Lander Models

In this chapter, an application scenario is presented for which we use the geometric properties and optimization techniques explained in previous chapters: the powered descent landing. It will be shown in the following pages that the theoretical constructs we have considered and derived throughout this thesis add significant value to real-world applications.
In the powered descent landing phase, a lander's thrusters are ignited to perform a soft landing on a celestial body after reducing the orbital speed with a parachute. These thrusters control the state vector

$$\begin{aligned} &\boldsymbol{x} : [t_0, t_f] \to \mathbb{R}^7, \\ &\boldsymbol{x}(t) = \begin{pmatrix} p_x(t), & p_y(t), & p_z(t), & v_x(t), & v_y(t), & v_z(t), & m(t) \end{pmatrix}^\top \end{aligned} \tag{5.1}$$

with $p_*(t), v_*(t) \in \mathbb{R}$, $* \in \{x, y, z\}$, describing the respective position and velocity component and $m(t) \in \mathbb{R}$ indicating the mass (implicitely the amount of fuel left) of the lander, subject to the following dynamical system:

$$\begin{pmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \\ \dot{p}_z(t) \end{pmatrix} = \begin{pmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{pmatrix} \tag{5.2a}$$

$$\begin{pmatrix} \dot{v}_x(t) \\ \dot{v}_y(t) \\ \dot{v}_z(t) \end{pmatrix} = -S(\omega)^2 \begin{pmatrix} p_x(t) \\ p_y(t) \\ p_z(t) \end{pmatrix} - 2S(\omega) \begin{pmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{pmatrix} + g(t) + \frac{T_c(t)}{m(t)}, \tag{5.2b}$$

$$\dot{m}(t) = -\gamma_m \|T_c(t)\|, \tag{5.2c}$$

$$\text{where } S(\omega) = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}.$$

Here, $\omega = \begin{pmatrix} \omega_1, & \omega_2, & \omega_3 \end{pmatrix}^\top$ is the target object's constant angular velocity vector. Besides the influence due to the celestial body's rotation, the acceleration of the lander depends on the gravity $g(t) \in \mathbb{R}^3$ and the thrust $T_c(t) \in \mathbb{R}^3$. The last term in (5.2b) implies a

reciprocally proportional relation between the acceleration and the mass. The change of mass is proportional to the magnitude of the thrust vector $T_c(t)$ with a proportionality factor $\gamma_m > 0$. The initial value m_0 and the dry mass m_{dry} represent box constraints for the state m:

$$m(t_0) = m_0, \quad m_{\text{dry}} \leq m(t) \leq m_0 \text{ for all } t \in \left[t_0, t_f\right]. \tag{5.3}$$

In both subsequently presented approaches, the first-order ordinary differential equation (5.2) with regard to (5.3) is further processed and simplified where required and then, the algorithms of Chapter 4 are applied. A full-discretization is performed on the optimal control problems based on the trapezoidal method. In the process, either NLPs or SOCPs have to be solved to approximate this Chapter's reachability and controllability sets.
WORHP (short for "**W**e **O**ptimize **R**eally **H**uge **P**roblems") solves NLPs, offering the option to use an SQP method or nonlinear interior-point method. It is notable that WORHP provides direct access to the parametric sensitivities [43]. Originally funded by the European Space Agency, WORHP has been developed at the University of Bremen [20]. A useful tool is TRANSWORHP, a software library to transcribe OCPs [41]. On the other hand, in this work, SOCPs are solved using ECOS (short for "**E**mbedded **Co**nic **S**olver"). ECOS relies on a primal-dual interior-point method with Nesterov-Todd scaling [53] and self-dual embedding [25]. Sensitivity derivatives can not be accessed in ECOS. This solver is the result of a collaboration between ETH Zurich and Stanford University.
All computations are performed on a Lenovo ThinkPad T14 Gen 1 notebook which promotes an AMD Ryzen 7 PRO 4750U with Radeon Graphics and 32 GB RAM. The processor has eight cores running two threads each. It features a base clock of 1.7 GHz and a maximum boost clock of 4.1 GHz.

5.1 Lunar Lander Model

The lunar lander model considers a spacecraft with a reaction control system, i.e., several thrust engines are available aside from a non-throttable main thruster to hypothetically push the lander in all directions. However, since the thrusters are stationary, the lander's attitude must be taken into account. Following [56], β and χ are introduced as pitch and yaw angles. In the following, (5.2) is simplified by neglecting the Moon's rotation, thus setting $\omega = 0$.

5.1.1 Transformation into dhc-frame

The position of the lander vessel can be specified based on downrange, altitude, and crossrange coordinates (dhc) by rotating the position vector at zero downrange, zero crossrange, and altitude h. The celestial body to land on is assumed to be spherical with radius r.
For the sake of clarity, we omit the time dependency of functions in the following derivation. As shown in Figure 5.1, first, a rotation around the x-axis with the angle $\gamma = \frac{c}{r}$, and

then, a rotation around the z-axis with the angle $\delta = \frac{d}{r}$ is performed:

$$\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = R_z(-\delta) R_x(\gamma) \begin{pmatrix} 0 \\ h+r \\ 0 \end{pmatrix}. \tag{5.4}$$

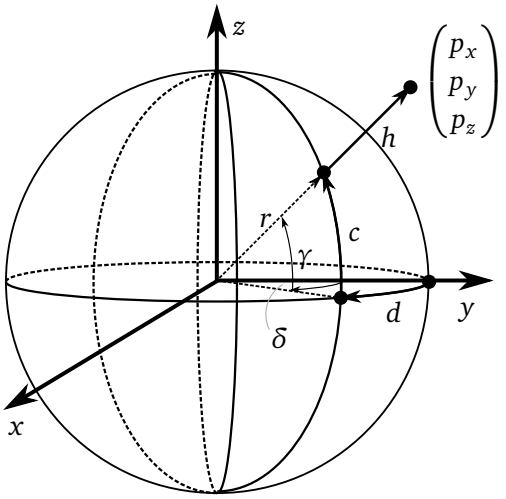


Figure 5.1: Relation between xyz- and dhc-coordinates

The gravitational field is assumed to be based on a spherical potential, thus the relation

$$g = R_z(-\delta) R_x(\gamma) \begin{pmatrix} 0 \\ -\frac{M \cdot G}{(r+h)^2} \\ 0 \end{pmatrix} \tag{5.5}$$

is obtained, where M and G denote the mass of the Moon and the gravitational constant. By transforming the total thrust vector $T = \left(-T_m - T_u, \quad -T_s, \quad -T_q\right)^T$ based on pitch and yaw angle, the effective thrust results

$$\begin{aligned} T_c &= R_z(\beta - \delta) R_y(\chi)\, T \\ &= \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} = \begin{pmatrix} \cos\left(\beta - \frac{d}{r}\right)\left((T_m + T_u)\cos\chi + T_q \sin\chi\right) - \sin\left(\beta - \frac{d}{r}\right) T_s \\ \sin\left(\beta - \frac{d}{r}\right)\left((T_m + T_u)\cos\chi + T_q \sin\chi\right) + \cos\left(\beta - \frac{d}{r}\right) T_s \\ -(T_m + T_u)\sin\chi + T_q \cos\chi \end{pmatrix}. \end{aligned} \tag{5.6}$$

Inserting the second derivative of (5.4) with regard to time on the left hand side and (5.5) and (5.6) on the right hand side of acceleration dynamic in (5.2) leads to

$$\begin{pmatrix} \ddot{d} \\ \ddot{h} \\ \ddot{c} \end{pmatrix} = \begin{pmatrix} \frac{r}{m(r+h)\cos\frac{c}{r}}\left(-T_1 \cos\frac{d}{r} + T_2 \sin\frac{d}{r}\right) + 2\dot{d}\left(\frac{\dot{c}}{r}\tan\frac{c}{r} - \frac{\dot{h}}{r+h}\right) \\ \frac{1}{m}\left[\left(-T_1 \sin\frac{d}{r} - T_2 \cos\frac{d}{r}\right)\cos\frac{c}{r} - T_3 \sin\frac{c}{r}\right] + \left[\left(\dot{d}\cos\frac{c}{r}\right)^2 + \dot{c}^2\right]\frac{r+h}{r^2} - \frac{M\dot{G}}{(r+h)^2} \\ \frac{r}{m(r+h)}\left[\left(T_1 \sin\frac{d}{r} + T_2 \cos\frac{d}{r}\right)\sin\frac{c}{r} - T_3 \cos\frac{c}{r}\right] - \frac{\dot{d}^2}{r}\sin\frac{c}{r}\cos\frac{c}{r} - \frac{2\dot{c}\dot{h}}{r+h} \end{pmatrix} \tag{5.7a}$$

$$\dot{m} = -\gamma_m |T_m| - \gamma_{RCS}\left(|T_u| + |T_s| + |T_q|\right). \tag{5.7b}$$

The state vector (5.1) is renewed as

$$\begin{aligned} &\boldsymbol{x}\colon \left[t_0, t_f\right] \to \mathbb{R}^9, \\ &\boldsymbol{x}(t) = \left(\dot{d}(t),\ \dot{h}(t),\ \dot{c}(t),\ d(t),\ h(t),\ c(t),\ \beta(t),\ \chi(t),\ m(t)\right)^\top. \end{aligned} \tag{5.8}$$

The pitch and yaw angles are continuous by adding

$$\dot{\beta} = \omega_\beta \text{ and } \dot{\chi} = \omega_\chi \tag{5.9}$$

to the system dynamic with $\omega_\beta, \omega_\chi$ being commanded angular rates. By specifying the control vector as

$$\begin{aligned} &\boldsymbol{u}\colon \left[t_0, t_f\right] \to \mathbb{R}^5, \\ &\boldsymbol{u}(t) = \left(T_u(t),\ T_s(t),\ T_q(t),\ \omega_\beta(t),\ \omega_\chi(t)\right)^\top. \end{aligned} \tag{5.10}$$

the differential equations of the lunar lander model can be written in the form

$$\dot{\boldsymbol{x}}(t) = f_{\text{dyn}}(\boldsymbol{x}(t), \boldsymbol{u}(t)). \tag{5.11}$$

A detailed derivation of the equations of motion can be found in [5].

5.1.2 Scenario and Results

The values for the constants of the equations of motion in (5.7) and box constraints for states (5.8) and controls (5.10) are taken from [5] and listed in Tables 5.1 and 5.2.

	symbol	**value**	**unit**
radius of Moon	r	1.737×10^6	m
mass of Moon	M	7.3490×10^{22}	kg
gravitational constant	G	6.6743×10^{-11}	$\mathrm{N\,m^2/kg^2}$
main thrust	T_m	1	-
fuel consumption rate by main engine	γ_m	5×10^{-4}	-
fuel consumption rate by RCS	γ_{RCS}	3.75×10^{-4}	-

Table 5.1: Constants used in the lunar lander model

The goal is to determine which values for downrange and crossrange are achievable at the end of a flight. The lander has an initial speed in the downrange direction and is losing altitude. The mass m and the thruster components T_u, T_s, T_q have no units due to scaling with the actual total initial mass of the lander. State m should not be less than the corresponding dry mass of $m_{\text{dry}} = 0.5$, indicating the amount of available fuel. The lower and upper bounds for the states and controls must be adhered to throughout the flight. At the landing point at time t_f, the lander should have no more velocities and be on the surface of the Moon ($h(t_f) = 0\,\text{m}$). A slight deviation of the fixed end states is tolerated:

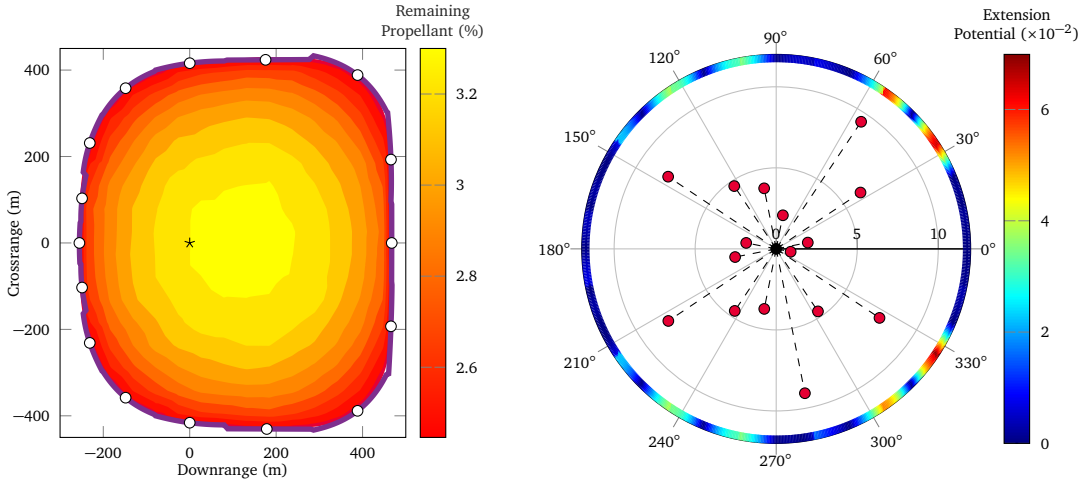
$$\Delta\boldsymbol{x}\left(t_f\right) = \left(1\,\text{m/s},\ 1\,\text{m/s},\ 1\,\text{m/s},\ \text{free},\ 1\,\text{m},\ \text{free},\ 10°,\ 180°,\ \text{free}\right)^\top. \tag{5.12}$$

We apply Algorithms D and G in this scenario. The results are visualized in the following two sections. A fixed and a free end time scenario are regarded, while $t_0 = 0$ is assumed.

	initial	final	lower bound	upper bound	unit
$\dot{d}$	5	0	$-\infty$	∞	m/s
$\dot{h}$	-19	0	$-\infty$	∞	m/s
$\dot{c}$	0	0	$-\infty$	∞	m/s
d	0	free	$-\infty$	∞	m
h	300	0	0	∞	m
c	0	free	$-\infty$	∞	m
β	-86	-90	-90	90	°
χ	0	free	-180	180	°
m	0.5397	free	0.5	0.5397	-
T_u	0	free	-0.222	0.222	-
T_s	0	free	-0.222	0.222	-
T_q	0	free	-0.222	0.222	-
ω_β	0	free	-2	2	°/s
ω_χ	0	free	-2	2	°/s

Table 5.2: Boundary values and box constraints of state vector and control vector of the lunar lander

5.1.2.1 Fixed End Time



(a) Set approximation

(b) Volume and approximation indicator

Figure 5.2: Reconstruction results of the lunar lander problem

Additionally to the aforementioned dynamics and constraints, a time of flight $t_f = 40\,\mathrm{s}$ is considered. Figure 5.2a shows the reachable set: It has roughly a trapezoidal shape with rounded corners. The set is almost symmetrical with regard to $y = 0$ and slightly shifted to the right. These characteristics are due to the initial velocities $\dot{d}(t_0) = 5$ and $\dot{c}(t_0) = 0$ in Table 5.2.

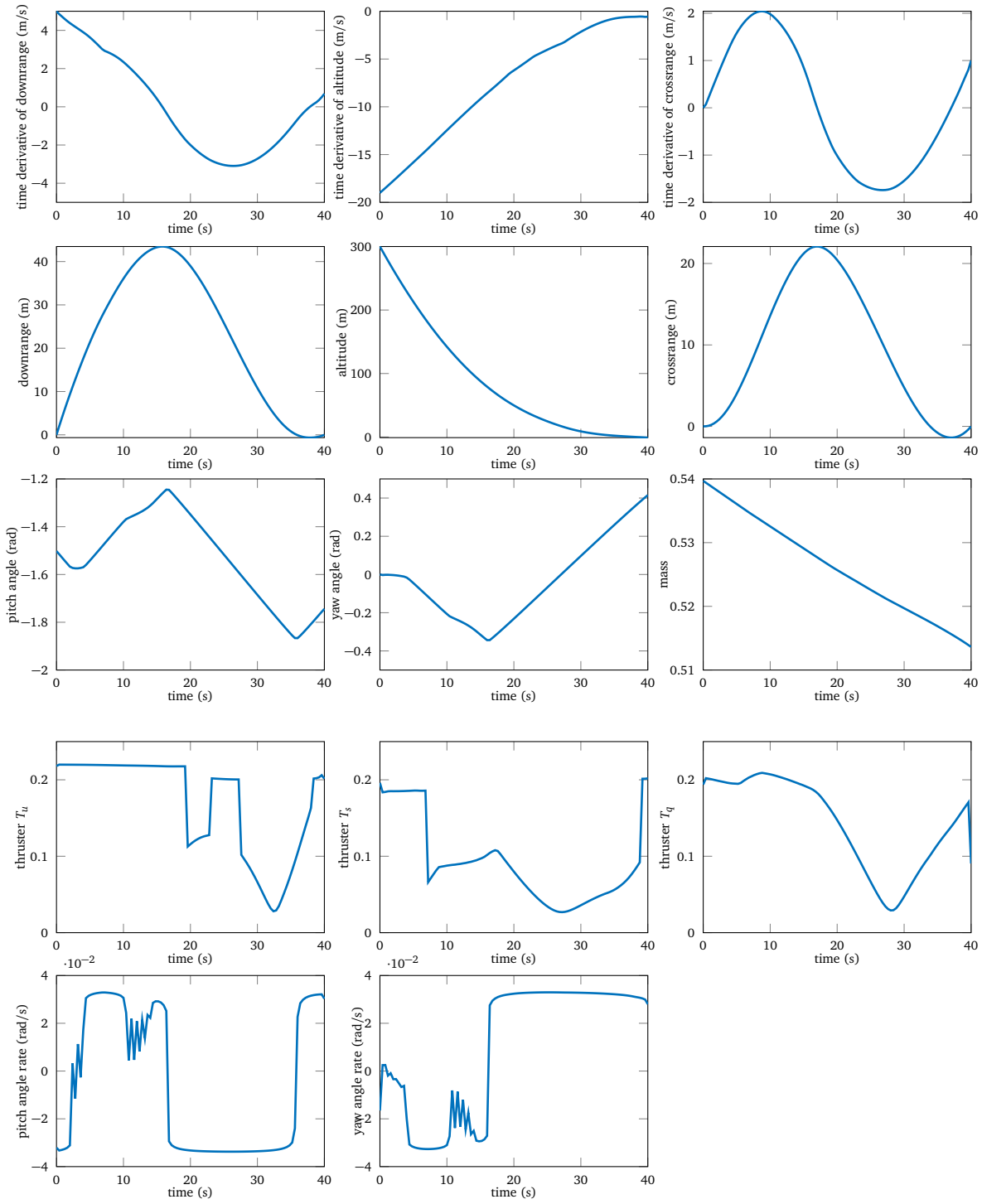


Figure 5.3: States and controls to reach the vantage point

The reachable downrange and crossrange states are within $[-255, 467]$ and $[-431, 431]$, respectively. The purple outline results from Algorithm G, while the white dots are the boundary elements found in the process. The black star marks the vantage point in $(0, 0)$. The figure also shows the outcome of the grid approach. A 20×20 grid is created from the interval $[-1000, 1000] \times [-1000, 1000]$, and a trajectory leading to the closest possible landing point is computed for each grid point.These trajectories also yield the remaining propellant at the end of the flight. Based on the remaining fuel, a color coding was additionally computed and visualized in Figure 5.2a. As the boundary of the reachable set is approached, the fuel consumption increases but the point where the propellant is completely used up is never reached under the given circumstances.
With the partition extension potential and the approximation discrepancy introduced in Definitions 4.21 and 4.22, Figure 5.2b is created. It indicates that the discontinuities in the transition between Taylor approximations of the boundary reconstructions are relatively low (less than 10 m), given the dimensions of the reachable set. Therefore, we can conclude that the interpolation based on 16 optimization runs yields a reasonable estimation of the shape of the reachable set. Most volume can be attained through additional optimizations along search directions near the rounded corners of the reachable set. However, the expected gain is relatively low.
Exemplary states and controls were computed to reach the black marked vantage point $o := (0, 0)^\top$ in Figure 5.2a illustrating the algorithms connection to optimization and optimal control. For this, G(o) was solved considering the dynamic system (5.7) and conditions formulated in Sections 5.1.1 and 5.1.2. Figure 5.3 shows the results. The added layer of derivatives for the control in (5.9) has lead to a smooth course of the curves representing the pitch and yaw angles.

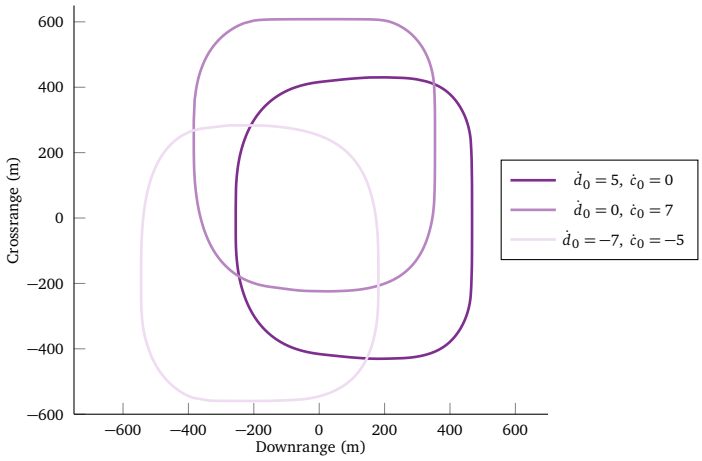


Figure 5.4: Reachable set of the lunar lander problem with varying initial velocities

Leaving the initial fall velocity at $\dot{h}(t_0) = -19$ and varying $\dot{d}_0$ and $\dot{c}_0$, Figure 5.4 depicts how the reachable set shifts. If the initial velocity in the crossrange direction is positive, and the initial downrange velocity is 0, the set shifts upward. An axis symmetry with respect to $x = 0$ can be recognized. If $\dot{d}(t_0) = -7$ and $\dot{c}(t_0) = -5$ are chosen, a large part of the set is in the third quadrant.

Figure 5.5 shows the outcome of a varied end time while remaining at the default scenario described in Table 5.2. It depicts the outlines of resulting reachable sets. The lighter the outline, the longer the time of flight t_f is. As expected, the area grows with the chosen end time.

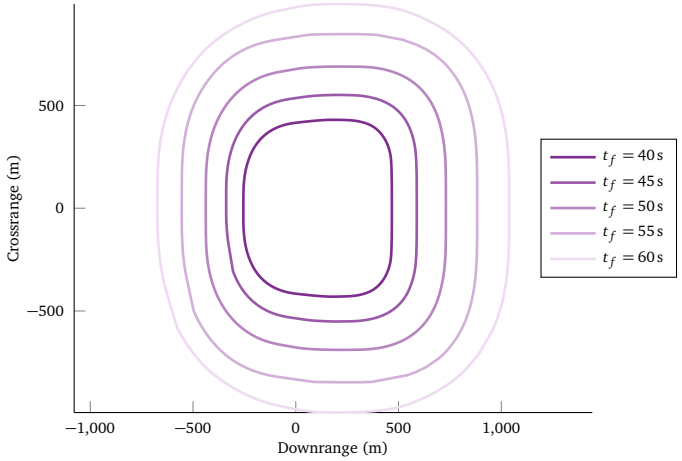
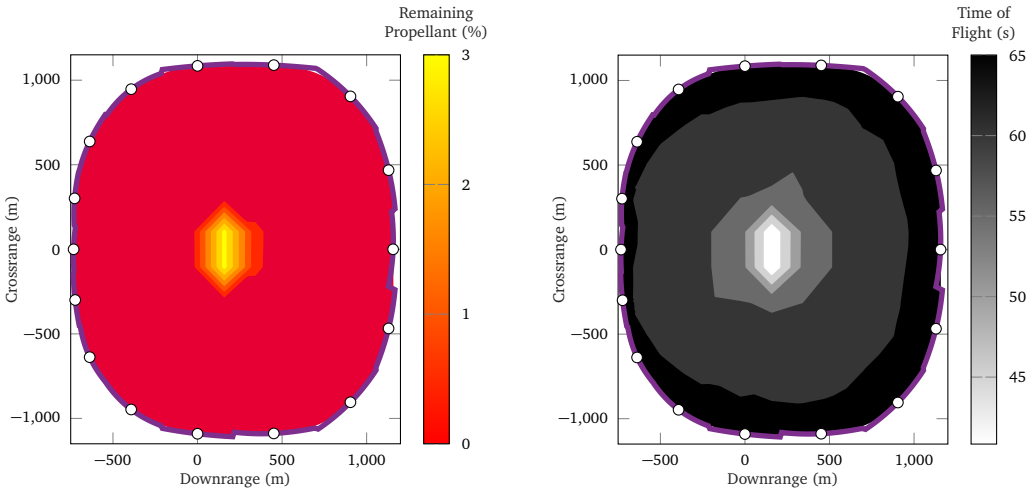


Figure 5.5: Reachable set of the lunar lander problem with varying end time

5.1.2.2 Free End Time



(a) Heatmap indicating remaining propellant

(b) Heatmap indicating end time

Figure 5.6: Reachable sets of the lunar lander problem with free end time

We repeat the evaluation done in Section 5.1.2.1 with a free end time. Figure 5.6 shows the reachable set under these circumstances. Compared to Figure 5.2a, it is larger with reachable downrange and crossrange coordinates ranging in $[-1099, 1099]$ and $[-1100, 1162]$, respectively. The effects of the additional degree of freedom are therefore directly visible.

The slight shift to more positive downrange coordinates is also visible in Figure 5.6. The input argument to apply Algorithm D is a 20×20 grid with equidistant points distributed over $[-1500, 2000] \times [-2000, 2000]$. Two heatmaps result: the first one concerns the remaining propellant (Figure 5.6a), and the other regards the flight time (Figure 5.6b). The fuel limits the time of flight. Most of the determined trajectories require all the available fuel, especially those close to the boundary. The time of flight ranges in $[40, 65]$. However, the optimization runs do not regularize the controls through an integral term of the Bolza problem formulation nor penalize the flight time. Therefore, the level transitions displayed in the heatmap are rather vague.

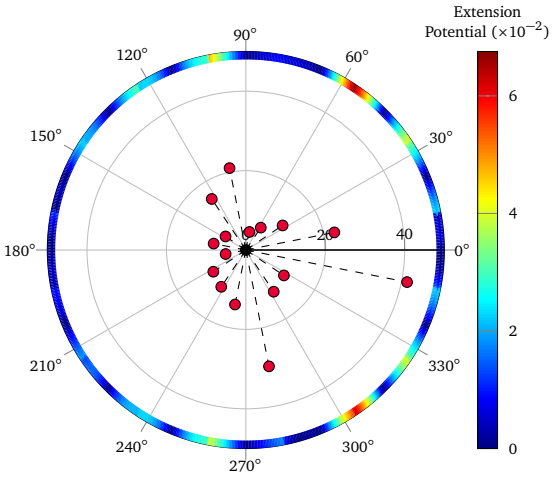


Figure 5.7: Volume and approximation indicator corresponding to Figure 5.6

Figure 5.7 shows that the boundary reconstruction has its most prominent discontinuity at 348.75° with an approximation discrepancy of 41 m. An additional expansion point could potentially improve the reconstruction. Similar to the fixed end time results, the largest volume potential lies in the rounded corners but is relatively small.

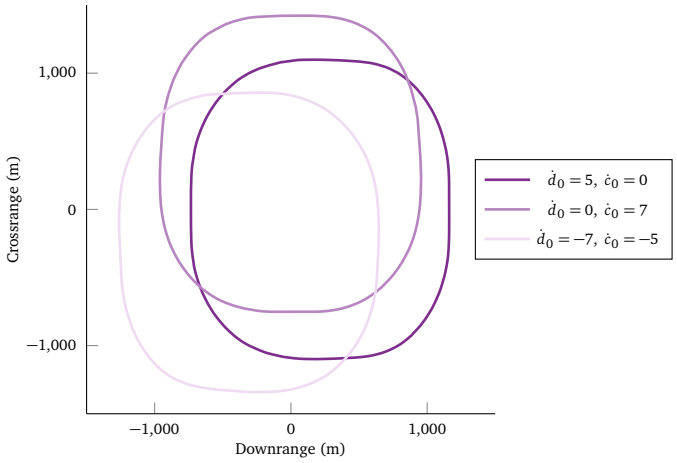
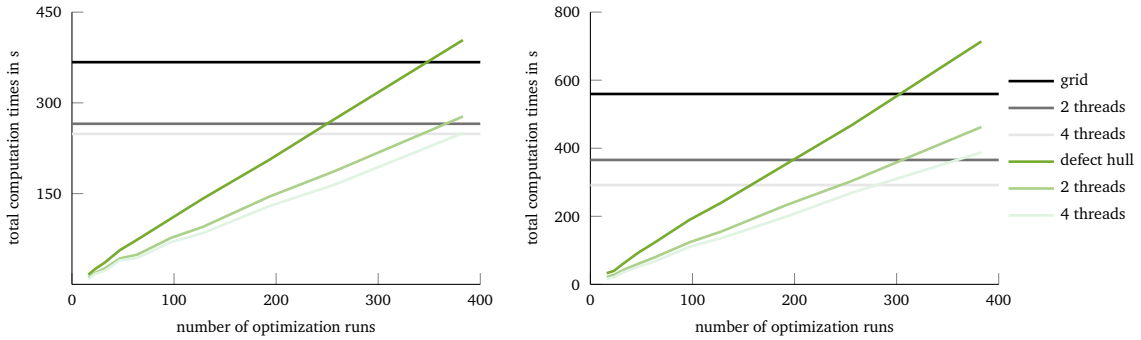


Figure 5.8: Reachable sets of the lunar lander problem with varying initial velocity and free end time

Figure 5.8 depicts the same effects as in Figure 5.4, when the initial velocities are varied. Again, the reachable set is shifted corresponding to the initial velocities in downrange and crossrange directions.

5.1.3 Computation Time

Computing the lunar lander reachability sets is a complex task involving solving a multitude of NLPs. Four hundred grid tasks and 16 defect hull tasks subject to the lunar lander constraints have been solved to obtain Figures 5.2a. The boundary reconstruction based on the defect hull output with 16 optimizations suffices to understand the rough shapes of the reachability sets. Those reconstructed shapes could become smoother when more optimization runs are performed. However, this smoothening comes with an additional computational cost. The green curves in Figure 5.9 show the course of the total computation times depending on the number of optimizations performed.



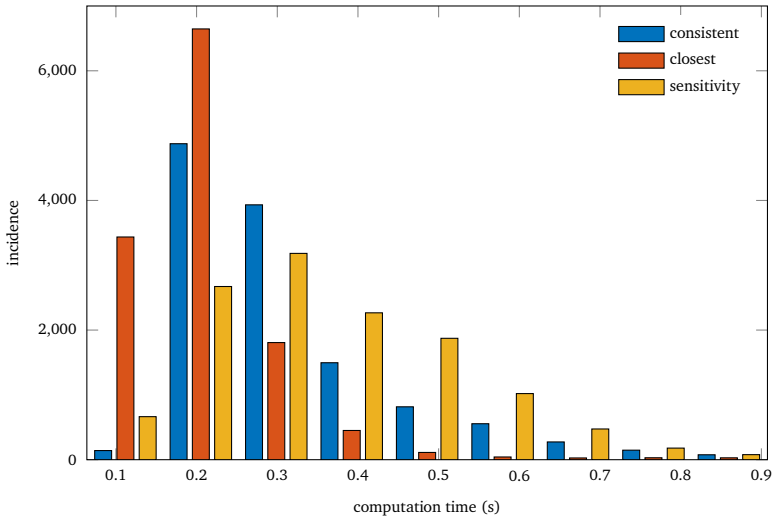
(a) Fixed end time **(b)** Free end time

Figure 5.9: Computation times to reconstruct the lunar lander reachability set

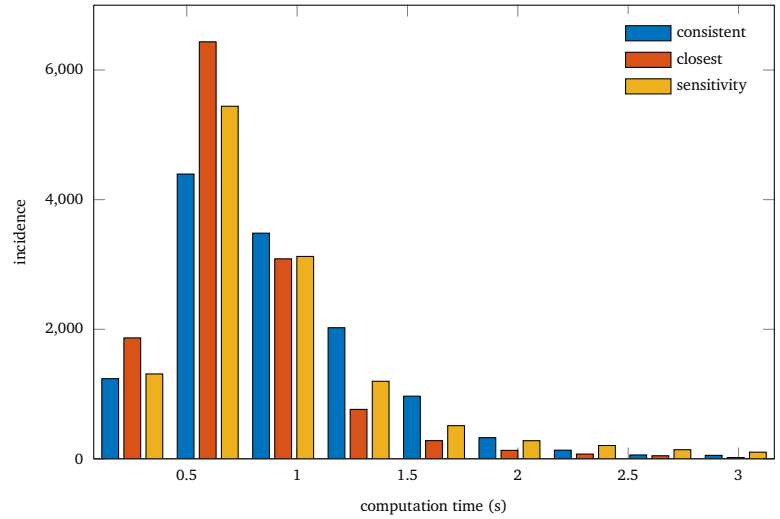
The increase of the green curves is almost linear. In addition, the computation times required for Algorithm D on a 20×20 grid are plotted in black and gray to compare the two methods. Both Algorithm G and Algorithm D are parallelizable. The corresponding curves in lighter colors show the computation times based on two and four threads. The computation time for the grid approach under said conditions can be reduced from 367.3 s to 265.4 s with two and 248.7 s with four threads. With free end time, the computation times drop from 559.5 s to 365.6 s and 291.8 s. In order to perform the defect hull approach with 16 optimizations, it takes 15.9 s, with two threads 11.2 s and four threads 8.7 s. With 383 optimizations, the computation times are 403.8 s, 277.4 s, and 250 s. The significant overhead with free end time is also noticeable in the defect hull approach. 16 and 383 optimizations take 32.9 s (21.4 s, 15.8 s) and 713.9 s (462.4 s, 387.8 s), respectively. The computing times do not scale according to the number of used threads because modern processor cores can overclock at low workloads, like a single-thread application [64]. The same optimization task can be solved noticeably faster when fewer processor cores in total are loaded.

The computation times for the grid approach in Figure 5.9 are on the order of minutes. Here, we achieved a significant improvement because, with the discretization approach in [5], the computation of reachable sets takes several days. To a small extent, this improvement can be attributed to the slightly improved technical prerequisites. From a numerical point of view, the discretization procedure with single-step methods presented in this work leads to significantly sparser internal matrices. Solvers that take the structures of matrices into account can immensely reduce the computational cost. Though the pseudo-spectral methods presented in [5] lead to smoother solutions, the internal matrices are dense. As a result, significantly more arithmetic operations have to be performed, which increases the computing time. Smooth controls are not needed to approximate the reachability set since the trajectories and controls found are not necessarily applied.
The great strength of the defect hull approach becomes especially apparent when only the approximate shape and magnitude of the reachable set are needed. With only 16 optimizations and a potential computation time of around 10 seconds, a good approximation of the set is obtained relatively quickly as illustrated in Figures 5.2a and 5.6a.

To conclude this part, the idea expressed in (4.3) is taken up again. Here, the question is treated of how the output from the grid algorithm can still be utilized. By the assignment (4.3), it is proposed to consider a grid point as a parameter which can be disturbed. Suppose a trajectory and control are sought to a landing point that is close to a grid point. In that case, the difference can be considered a perturbation, and the result of the Taylor approximation based on sensitivities can be used as an initial guess. Besides this strategy, the more straightforward idea is to use as an initial guess the result of the discretized optimal control problem that led to the trajectory of the closest grid point. Both approaches are compared to the classical approach, with only a consistent initial guess for any potential landing point. Over 10000 random landing points have been chosen respectively for the fixed and free end time scenario. Trajectories are obtained by solving grid tasks $G(\hat{p})$ in which all landing points can be inserted as $\hat{p}$.
Figure 5.10a shows the result for the lunar lander problem and indicates the incidences subject to the computation times. In addition to that, further details like the 95th percentile have been separately determined. It becomes clear that the superior initial guess approach is the one choosing the initial guess given by the closest grid points. Significantly more trajectories can be found in about 0.2 s or less. 95 % of the optimizations are solved in less than 3.8 s, while the 95th percentile using a consistent initial guess is 0.73 s. The incidences provided the sensitivity based initial guesses are more distributed over the illustrated time frame. 95 % of the optimizations are performed in less than 0.71 s. Therefore, the latter hints only at a minor improvement to the consistent initial guess strategy. Similar results can be observed in Figure 5.10b. The 95th percentile for each approach is 1.8 s (consistent), 1.4 s (closest) and 2.5 s (sensitivity). These numbers are also indicators that, in general, the computation time increases greatly when an OCP with free end time is regarded.
In conclusion, the sensitivity-based initial guess strategy proposed in (4.3) does not represent an improvement to the classic, consistent initial guess strategy. However, choosing a trajectory and controls that lead to the closest grid point reduces the computation times



(a) Fixed end time



(b) Free end time

Figure 5.10: Computation times to get trajectories based on Algorithm D and different initial guess strategies

and makes the solving procedures more robust. Therefore, it can be concluded that a seemingly closer trajectory and controls with infeasibilities due to the first-order Taylor approximation cannot outweigh the actual feasibility of an initial guess when aiming for convergence.

5.2 Fully Constrained Marslander Model

In [28], Eren et al. present a Mars landing scenario that, through relaxation, leads to a trajectory generation approach based on second-order cone programming. Besides the

equation of motion in (5.2), the following constraints are regarded

$$\|v(t)\| \leq V_{\max} \tag{5.13a}$$
$$\|E(p(t)-p(t_f))\| \leq c^T(p(t)-p(t_f)) \tag{5.13b}$$
$$0 < \rho_1 \leq \|T_c(t)\| \leq \rho_2 \tag{5.13c}$$
$$\hat{n}\frac{T_c(t)}{\|T_c(t)\|} \geq \cos(\theta) \tag{5.13d}$$

for all $t \in [t_0, t_f]$. The inequality in (5.13a) is a speed limit, so the velocities are bounded over time $t \in \left[t_0, t_f\right]$. The expression (5.13b) is the so-called glideslope constraint that restricts the position components to be in a cone. Due to this constraint, the flight path does not become too shallow or cross the surface. As in [28], we choose

$$c = \frac{1}{\tan\left(\gamma_{\text{gs}}\right)}\begin{pmatrix}1\\0\\0\end{pmatrix}, \quad E = \begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\end{bmatrix}. \tag{5.14}$$

There is a natural upper bound for the thrust due to an engine's performance expressed by $\rho_2 \geq 0$ in (5.13c). The lower bound $\rho_1 \geq 0$ for the norm avoids combustion instability issues for small thrusts (cf. [50], p. 8). Providing a normalized vector $\hat{n} \in \mathbb{R}^3$, (5.13d) ensures that a vehicle's thruster points correctly. The angle between a thruster's pointing and the desired direction may deviate by a tolerated angle $0 \leq \theta \leq \pi$. Finally, boundary conditions for the mass are considered:

$$m(t_0) = m_0, \qquad m(t_f) \geq m_f \tag{5.15}$$

5.2.1 Convexification

While (5.13a), (5.13b), and the upper bound in (5.13c) are second-order cone constraints, some non-convexities forbid incorporating the discretized version of the dynamics (5.2) and constraints (5.13) as a feasibility problem into an SOCP. These non-convexities are located in the thruster term in the acceleration equation (5.2c) (division by mass), the negative norm in the mass depletion (5.2c), in the lower bound for the thrust magnitude (5.13c), and the thrust pointing constraint (5.13d). Following the approach in [9], the relaxation starts by introducing a slack variable $\Gamma(t) \in \mathbb{R}, t \in \left[t_0, t_f\right]$ to replace (5.13c) and (5.13d) with

$$\|T_c(t)\| \leq \Gamma(t), \tag{5.16a}$$
$$\hat{n}T_c(t) \geq \Gamma(t)\cos(\theta), \tag{5.16b}$$
$$\rho_1 \leq \Gamma(t) \leq \rho_2. \tag{5.16c}$$

While (5.16a) is a second-order cone constraint, (5.16b) and (5.16c) are linear inequalities. Thus, (5.16) displays convex constraints. Note that if (5.16a) is active, the original non-convex conditions (5.13c) and (5.13d) are fulfilled. In this case, the relaxation is termed

lossless convexification in [50]. We establish variables and substitute as follows to further remove the non-linearities in the dynamics:

$$\sigma(t) := \frac{\Gamma(t)}{m(t)}, \quad u(t) := \frac{T_c(t)}{m(t)}, \quad z(t) := \ln m(t) \tag{5.17}$$

Hence, the constraint (5.16c) changes to

$$\rho_1 e^{-z(t)} \leq \sigma(t) \leq \rho_2 e^{-z(t)}, \tag{5.18}$$

where this time, the upper bound describes a non-convex region. We can bypass this problem by using a working point

$$z_0(t) = \ln(m_0 - \gamma \rho_2 t), \tag{5.19}$$

to estimate $e^{-z(t)}$ with a first-order Taylor approximation[1] for the upper bound. For the lower bound, we use a second-order approximation:

$$\rho_1 e^{-z_0(t)}\left[1-(z(t)-z_0(t))+0.5(z(t)-z_0(t))^2\right] \leq \sigma(t) \leq \rho_2 e^{-z_0(t)}[1-(z(t)-z_0(t))]. \tag{5.20}$$

Using the remainder terms of the Taylor expansions, it can be shown that these approximations of $e^{-z(t)}$ are sufficiently accurate (compare [9], p. 1359). The changes due to substitution (5.17) in the dynamics and other constraints are more intuitive and can be found in the following summary of the relaxation:

$$\begin{pmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \\ \dot{p}_z(t) \end{pmatrix} = \begin{pmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{pmatrix} \tag{5.21a}$$

$$\begin{pmatrix} \dot{v}_x(t) \\ \dot{v}_y(t) \\ \dot{v}_z(t) \end{pmatrix} = -S(\omega)^2 \begin{pmatrix} p_x(t) \\ p_y(t) \\ p_z(t) \end{pmatrix} - 2S(\omega) \begin{pmatrix} v_x(t) \\ v_y(t) \\ v_z(t) \end{pmatrix} + g + u(t), \tag{5.21b}$$

$$\dot{z}(t) = -\gamma_m \sigma(t), \tag{5.21c}$$

$$\|v(t)\| \leq V_{\max}, \tag{5.21d}$$

$$\|E(p(t)-p(t_f))\| \leq c^T(p(t)-p(t_f)), \tag{5.21e}$$

$$\|u(t)\| \leq \sigma(t), \tag{5.21f}$$

$$\cos(\theta)\sigma(t) \leq \hat{n}u(t), \tag{5.21g}$$

$$\begin{aligned} \rho_1 e^{-z_0(t)}\left[1-(z(t)-z_0(t))+0.5(z(t)-z_0(t))^2\right] &\leq \sigma(t) \\ &\leq \rho_2 e^{-z_0(t)}[1-(z(t)-z_0(t))] \end{aligned} \tag{5.21h}$$

$$z(t_0) = \ln m_0, \qquad z(t_f) \geq \ln m_f \tag{5.21i}$$

[1] These approximations are necessary to ultimately incorporate both inequalities into an SOCP. Strictly speaking, however, one must note that the linearization of the upper bound in (5.20) is a tightening of the original constraint and not a relaxation. In fact, due to the strict convexity of the composition $\exp(-z(t))$, $\exp(-z(t)) > \exp(-z_0(t))(1-(z(t)-z_0(t)))$ holds true.

The state vector for the relaxed problem is

$$\begin{aligned} &\boldsymbol{x}\colon \left[t_0, t_f\right] \to \mathbb{R}^7, \\ &\boldsymbol{x} = \begin{pmatrix} p_x(t), & p_y(t), & p_z(t), & v_x(t), & v_y(t), & v_z(t), & z(t) \end{pmatrix}^\top. \end{aligned} \tag{5.22}$$

and the control vector is

$$\begin{aligned} &\boldsymbol{u}\colon \left[t_0, t_f\right] \to \mathbb{R}^4, \\ &\boldsymbol{u}(t) = \begin{pmatrix} u_1(t), & u_2(t), & u_3(t), & \sigma(t) \end{pmatrix}^\top. \end{aligned} \tag{5.23}$$

5.2.2 Lossless Convexification

It is important to note all efforts to obtain (5.21) has lead to a non-equivalent variation of the original problem (5.2) with (5.13). For fixed boundary values

$$\begin{aligned} p(t_0) &= p_0, \quad v(t_0) = v_0 \\ p(t_f) &= p_f, \quad v(t_f) = v_f \end{aligned} \tag{5.24}$$

a lossless convexification result exists for the energy optimal problem

$$\min_{t_f, \boldsymbol{x}, m, T_c, \Gamma} \int_{t_0}^{t_f} \Gamma(t) \, \mathrm{d}\, t \tag{5.25}$$

subject to (5.2), (5.13a), (5.13b), (5.15), (5.16), (5.24).

The solution $\left\{t_f^*, \boldsymbol{x}^*, m^*, T_c^*, \Gamma^*\right\}$ of (5.25) also yields minimizer $\left\{t_f^*, \boldsymbol{x}^*, m^*, T_c^*\right\}$ of

$$\min_{t_f, \boldsymbol{x}, m, T_c} \int_{t_0}^{t_f} \left\| T_c(t) \right\| \mathrm{d}\, t \tag{5.26}$$

subject to (5.2), (5.13), (5.24)

(cf. Theorem 1 in [21]). The solution of

$$\min_{t_f, \boldsymbol{x}, \boldsymbol{u}} \int_{t_0}^{t_f} \sigma(t) \, \mathrm{d}\, t \tag{5.27}$$

subject to (5.21), (5.24)

always implies a feasible solution of (5.26) after back-substitution [21]. There are two aspects to consider here. First, the time of flight t_f is an optimization variable in (5.25), (5.26), and (5.27). Thus, the discretization of those optimal control problems does not result in a SOCP or CP. Instead, the (5.27) is solved several times with different fixed $t_f \in (t_{\min}, t_{\max}]$ until the minimum cost is determined. Examples of interval limits are

$$t_{\min} = 0 \text{ and } t_{\max} = \frac{m_0 - m_f}{\gamma_m \rho_1}.$$

Due to (5.21c), (5.27) can thereby be converted into a Mayer-problem through following equivalence:

$$\int_{t_0}^{t_f} \sigma(t)\,\mathrm{d}\,t = -\frac{1}{\gamma_m}\left(z(t_f) - z(t_0)\right).$$

Secondly, the result of a grid, online convex hull, or defect hull task subject to the relaxed problem (5.21) for a fixed time of flight is not necessarily feasible in terms of the original dynamics (5.2) and constraints (5.13). These tasks have their respective objective functions but the lossless convexification conclusion is coupled to a cost functional that minimizes energy. However, due to convexity of the relaxed feasible set, any point in the forward or backward reachability set associated with (5.21) is necessarily linked to some feasible state trajectory $\boldsymbol{x}$ and controls $\boldsymbol{u}$ for time $t \in \left[0, t_f\right]$. Thus, the solution set of (5.27) cannot be empty and an energy-minimal solution must exist with an optimal time of flight t_f^*.

5.2.3 Scenario and Results

In this section, the relaxed reachability and controllability set for the Mars lander scenario are approximated using the online convex hull and defect hull algorithm. The results of both algorithms are then compared. Table 5.3 lists the values for the constants used in the dynamic model and the constraints. They are taken from [28].

	symbol	**value**	**unit**
Martian gravity vector	g	$\begin{pmatrix}-3.71, & 0, & 0\end{pmatrix}^\top$	$\mathrm{m/s^2}$
Martian rotation vector	ω	$\begin{pmatrix}2.53, & 0, & 6.62\end{pmatrix}^\top$	$10^{-5}\,\mathrm{rad/s}$
wet mass of lander	m_0	2000	kg
dry mass of lander	m_{dry}	1700	kg
maximum thrust capacity	$T_{\max}$	24	kN
lower bound for thrust	ρ_1	$0.2 \cdot T_{\max}$	kN
upper bound for thrust	ρ_2	$0.9 \cdot T_{\max}$	kN
fuel consumption	γ_m	5×10^{-4}	s/m
maximum thrust pointing deviation	θ	45	°
glideslope angle	γ_{gs}	15	°
maximum velocity	$V_{\max}$	130	m/s
time of flight	t_f	50	s

Table 5.3: Values for constants used in the Mars lander model

For the controllability set, initial velocities

$$v(t_0) = \begin{pmatrix}-100 & 20 & 0\end{pmatrix}^\top \tag{5.28}$$

are assumed and for those, feasible initial positions ought to be determined such that the final states

$$p(t_f) = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^\top, \quad v(t_f) = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^\top \tag{5.29}$$

can be reached. For the reachability analysis, the initial states

$$p(t_0) = \begin{pmatrix} 3000 & 1000 & 500 \end{pmatrix}^\top, \quad v(t_0) = \begin{pmatrix} -80 & 20 & 0 \end{pmatrix}^\top \tag{5.30}$$

are fixed and the goal is to find all positions at which the velocity can be reduced to 0 after a fixed time of flight t_f:

$$v(t_f) = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^\top. \tag{5.31}$$

For both scenarios, the initial value m_0 and the lower bound m_{dry} are considered for the mass of the spacecraft. The start time $t_0 = 0$ is assumed. Figures 5.11 and 5.12 illustrate the computation times for the online convex hull and the defect hull. The respective histograms and scatter plots exhibit times to perform 2980 optimizations runs[2]. Both the controllability and reachability scenario are examined. The colored lines in both scatter plots indicate the median and the 95th percentile. Despite additional constraints in the defect hull algorithm, the calculation effort of the algorithms does not differ much and ranges in a very similar order in terms of computation time. The median of the defect hull tasks is 0.009 38 s and marginally lower than the median of the online convex hull tasks (0.009 447 s). Conversely, 95% of the online convex hull tasks are done after 0.012 721 s, while this value is 0.012 967 s for the defect hull algorithm. These computation times may be affected by different processor loads or unclean programming, among other reasons, but it suggests that the effort required to optimize the different problems is comparable.
The following two subsections treat the controllability and reachability results for the Mars lander separately. In the process, we examine the outcome of the online convex hull and defect hull algorithms. The conjecture in [28] that the relaxed reachability set equals the original set has neither been shown nor disproved in this work. The same holds for the relaxed and original controllability set. However, for both scenarios, more than 15,000 elements of an approximate relaxed set have been checked to determine whether trajectories exist that have those elements as boundary values and fulfill the corresponding original conditions. Since the feasibility problem is nonlinear, an NLP solver must be applied for these checks. Nearly all elements have been admissible as initial or final states with deviations of less than one centimeter. The most prominent outliers have a deviation of under one meter. Individual adjustments to the parameters of the NLP solver could probably mitigate these more significant deviations. Due to the total number of elements that need an inspection, this has not been done. In addition, an experiment is conducted related to Corollary 4.15. Each element of an approximating polytope of the defect hull algorithm

[2]Both the online convex hull and the defect hull algorithm have been applied with $n_\mathcal{V}$ optimizations, where $n_\mathcal{V} \in \{16, 23, 32, 47, 64, 97, 128, 193, 251, 256, 383\}$ is an element of a set of powers of 2 and prime numbers. These numbers add up to 1490 which multiplied by the number of scenarios (reachability and controllability) makes 2980.

can be assigned a feasible trajectory and a control subject to the relaxed problem (5.20) since the dynamics and constraints are convex. Therefore, we shall examine whether an adaptive initial guess through Corollary 4.15 is more suitable than a consistent nominal one. The consistent initial guess is chosen as the solution of G(o) subject to (5.21), where o is the vantage point for the defect hull algorithm.

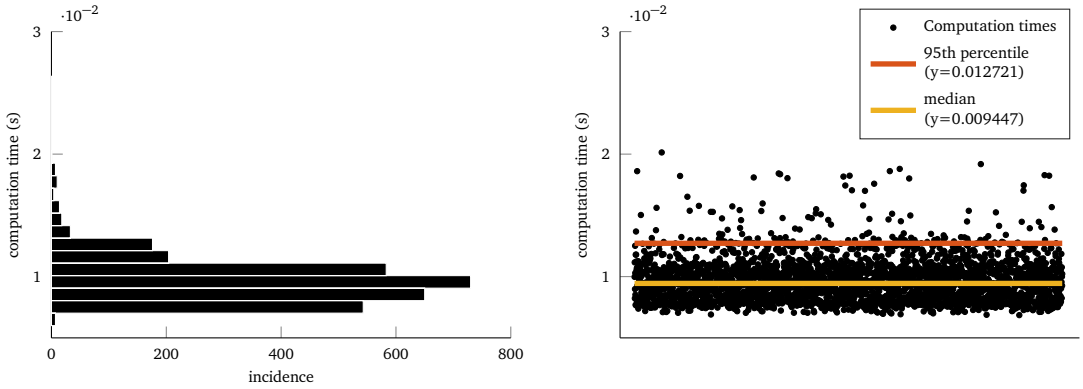


Figure 5.11: Overview of computation times for online convex hull tasks

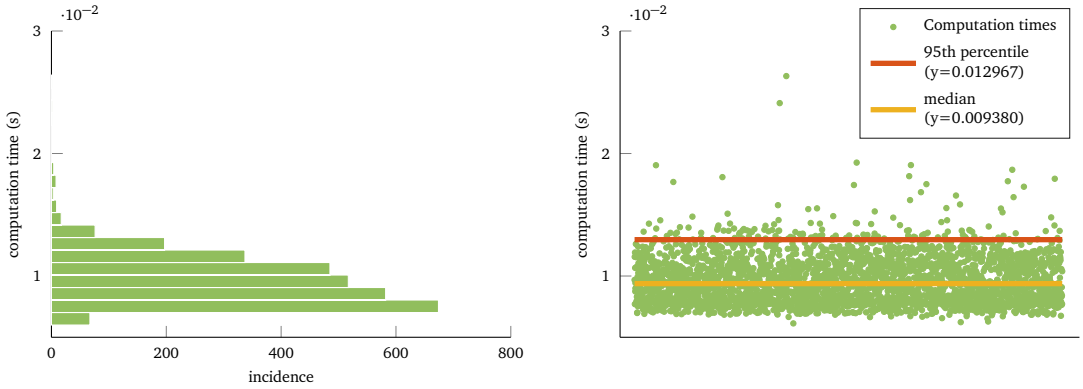
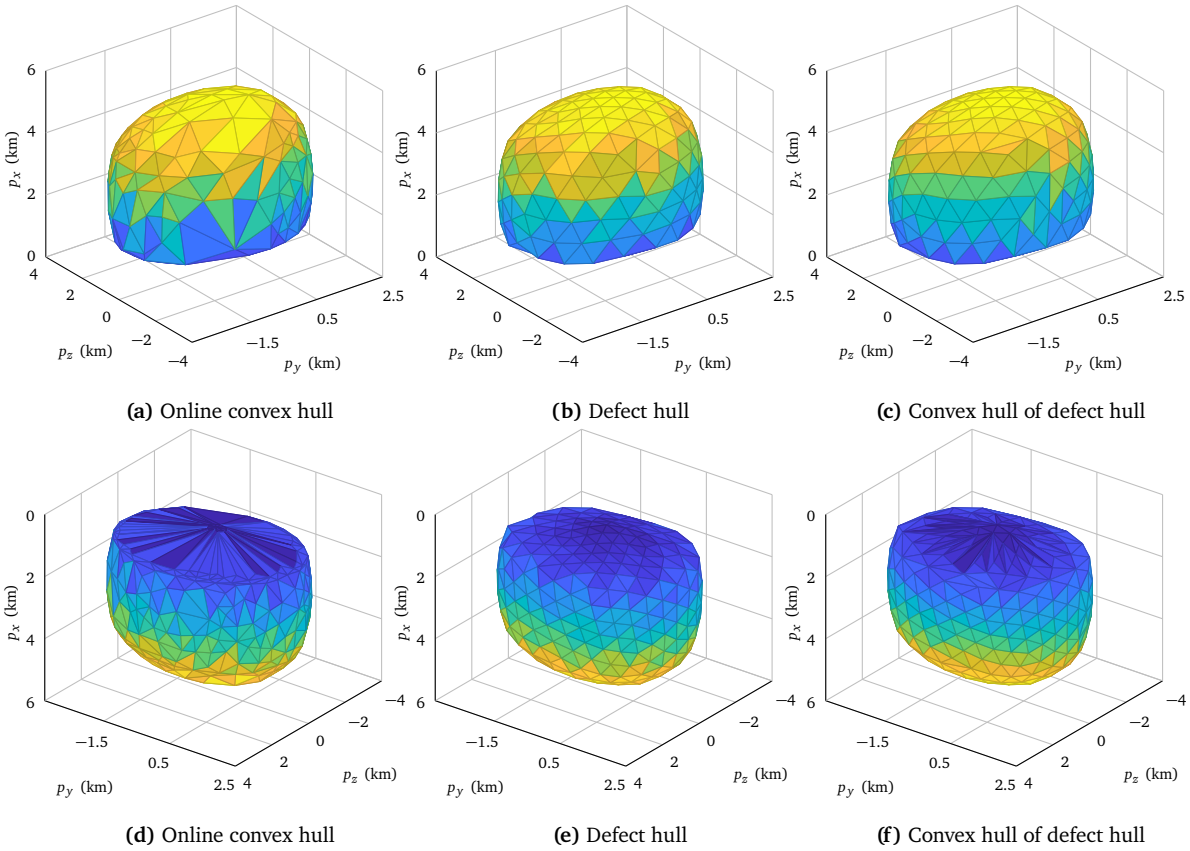


Figure 5.12: Overview of computation times for defect hull tasks

5.2.3.1 Controllability

Figure 5.13 illustrates polytopes with 256 vertices that approximate the relaxed controllability set of the Mars lander. The first row shows the respective results of the online convex hull algorithm (5.13a), the defect hull (5.13b), and the convex hull of the inner polytope (5.13c) that resulted from Algorithm G. The second row exhibits the same results from the upside-down view. A slight shift to negative p_y values is visible and caused by the positive initial velocity in p_y-direction in (5.28). Compared to the defect hull, the product from the online convex hull algorithm has a less regular triangulation but more distinctive shapes. Apparently, the lower end converges like a cone in the actual relaxed controllability set. This is observable in Figure 5.13d but is not happening in Figure 5.13e. However, applying

the convex hull on the vertices of the defect hull, the correction of the triangulation seems to become necessary just there, as Figure 5.13f shows. The convex hull shows a similar boundary description on the upper part of the defect hull approximation in Figures 5.13b and 5.13c. Again, it can be concluded that the defect hull algorithm produces good results, especially for smooth surfaces. For hard edges, the online convex hull approach is more suitable.

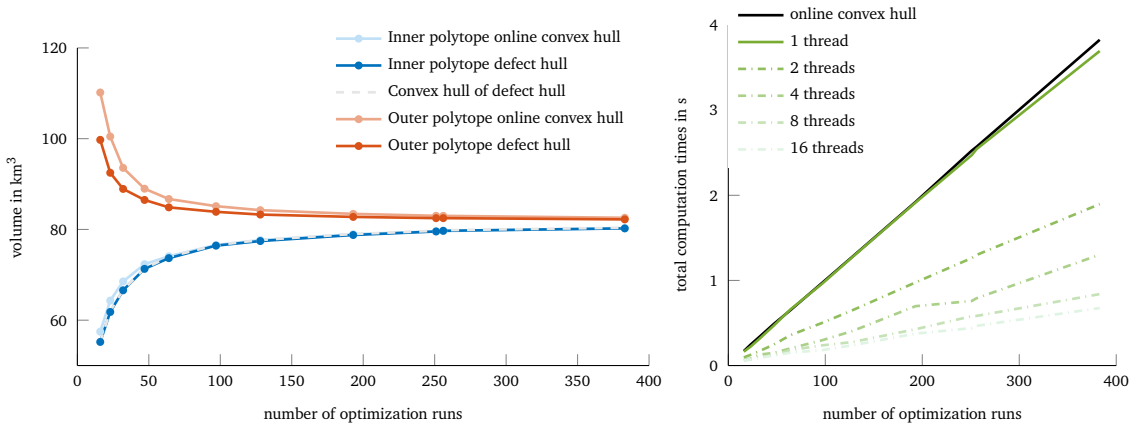


(a) Online convex hull **(b)** Defect hull **(c)** Convex hull of defect hull

(d) Online convex hull **(e)** Defect hull **(f)** Convex hull of defect hull

Figure 5.13: Visualized approximations of the relaxed controllability set

Figure 5.14 summarizes all approximations of the controllability set. In Figure 5.14a, the volumes of the generated polytopes are plotted against the number of performed optimizations. The convergence behavior of the curves is recognizable the larger the values of the x-axis become. The online convex hull algorithm covers a larger area of the sought set with few optimizations. However, with an increasing number of optimization runs, the defect hull algorithm is in no way inferior. The inner polytope of the defect hull algorithm is always nearly congruent to its convex hull and covers more than 99% of the space. This can be seen from the almost identical gray and dark blue curves in Figure 5.14a. The mirrored image is recognizable concerning the outer polytope. The upper approximation, which arises with the defect hull, is closer to the sought relaxed controllability set than the online convex hull as the volumes are consistently lower. The total computation times are plotted



(a) Course of volumes

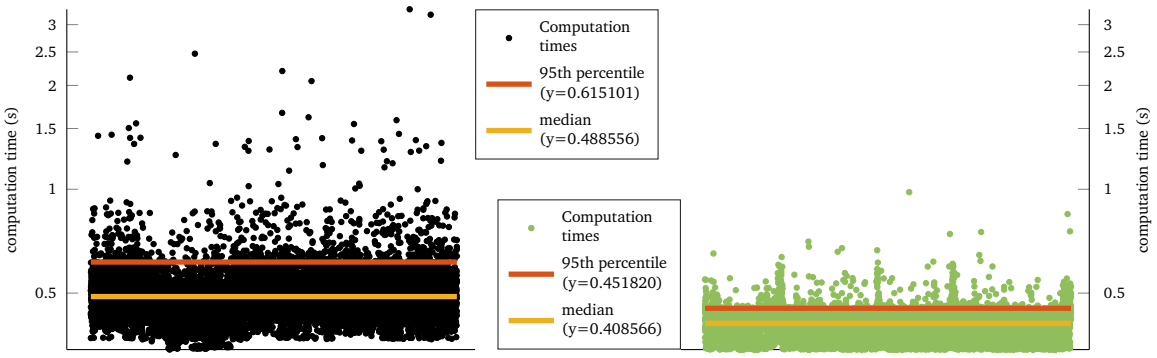
(b) Total computation times

Figure 5.14: Summarized results of the controllability scenario

against the number of optimization runs in in Figure 5.14b. The defect hull approach is slightly faster, and it can be further accelerated by distributing the computational load on multiple threads. However, the computation time does not quite scale accordingly because the effort to distribute the tasks and resources increases with the number of threads. Nevertheless, the added value of a parallel calculation is significant. It takes 2.57 s to construct the polytope in Figure 5.13a, whereas the polytope in Figure 5.13b needs 2.54 s or 1.3 s, 0.79 s, 0.58 s and 0.47 s as the number of threads doubles.

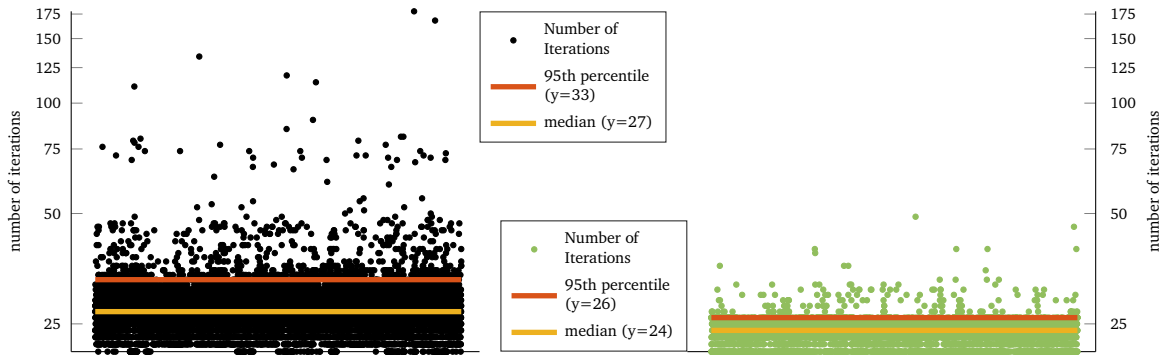
Over 15,000 initial positions that are admissible subject to the relaxed problem are checked to see if they are part of the controllability set of the original problem. Random elements $\hat{z}$ from the inner defect hull approximation of the relaxed controllability set after 256 optimizations are considered as target points to be best approximated by a solution of $G(\hat{z})$ subject to the nonlinear constraints (5.2), (5.13), and boundary values (5.28), (5.29). In the process, feasible trajectories and controls are found that feature a respective objective value of nearly 0. This is seen as an indicator that the relaxed and the actual controllability set coincide for this specific scenario, as the conjecture in [28] states. Simultaneously, this experiment is conducted from another viewpoint. It yields insights into how long it takes to compute trajectories starting from different initial states that lead to a common final state. Recalculation of a reference trajectory is needed in a landing scenario if the initial state has been inadequately estimated. This experiment offers good prerequisites to compare the two initial guess strategies. The adaptive initial guess generation exploits the results of the relaxed controllability approximation. It is superior compared to a classic "one for all" initial guess in terms of computation time, as it turns out.
As Figure 5.15 indicates, the median of the necessary computation times lies at about 0.49 s using a consistent initial guess, while this value is 0.41 s when adaptive initial guesses are applied. Both values have a relative difference of about 16%. The 95th percentile can be even more decreased. From about 0.62 s to 0.45 s, a reduction by more than 26% is possible.
Figure 5.16, reflects the shortened optimizations through the number of necessary itera-

tions in the solution processes. While the classical strategy leads to a median of 27 iterations and a 95th percentile at 33 iterations, the median in the evaluation of the adaptive strategy is 24 iterations, and the corresponding percentile is 26 iterations. This means a decrease of more than 11% for the median and more than 21% for the 95th percentile. In both Figures 5.15 and 5.16, the significant reduction of the respective 95th percentile is visible as there are no samples beyond 1 s or 50 iterations. Also, the median and the 95th percentile are closer together when adaptive initial guesses are used.



(a) Consistent initial guess **(b)** Adaptive initial guess

Figure 5.15: Computation times using different initial guess strategy in the controllability scenario



(a) Consistent initial guess **(b)** Adaptive initial guess

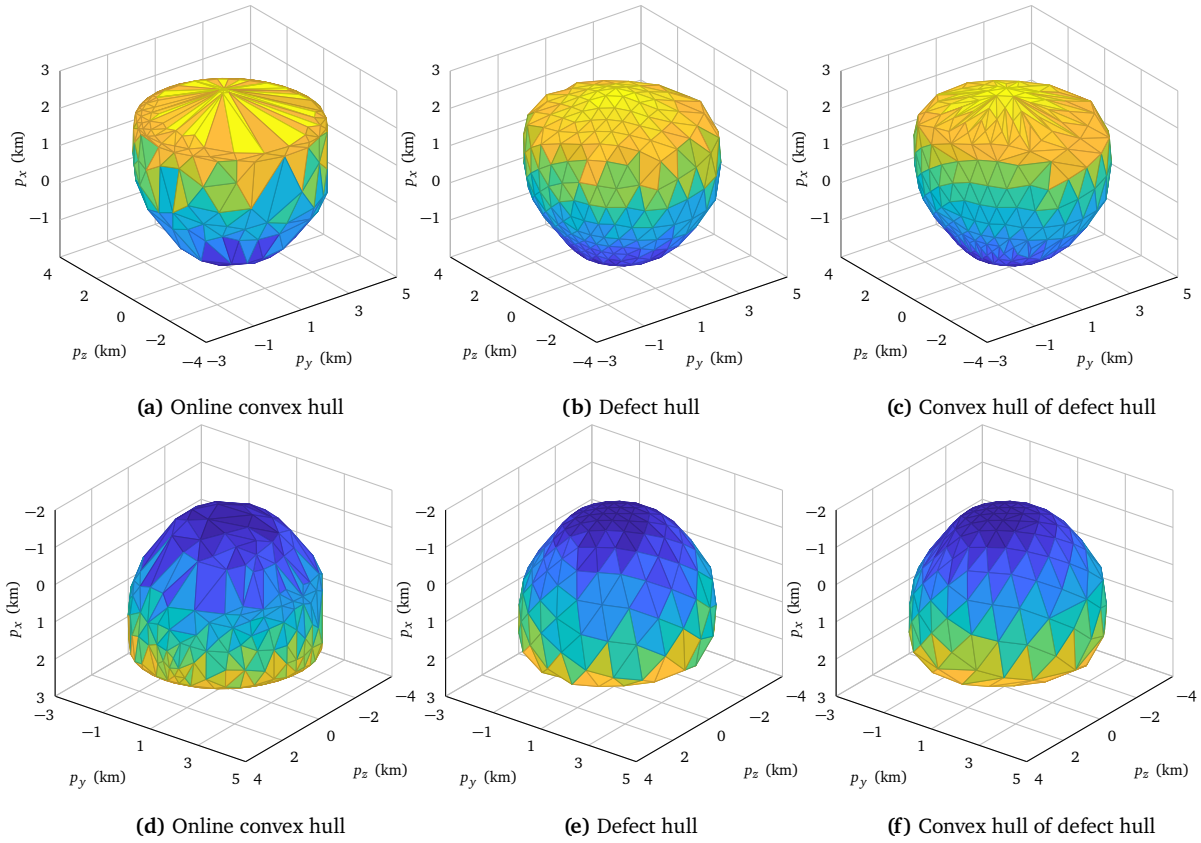
Figure 5.16: Number of iterations using different initial guess strategy in the controllability scenario

A precomputed relaxed controllability set can significantly speed up the recalculation of a reference trajectory and provide confidence that a particular end state can still be achieved. Controls and state trajectories found during optimizations of the relaxed problem can thus be effectively reused in the nonlinear context.

5.2.3.2 Reachability

Figure 5.17 illustrates polytopes with 256 vertices approximating the relaxed reachability set of the Mars lander. The first row shows the respective results of the online convex

hull algorithm (Figure 5.17a), the defect hull (Figure 5.17b), and the convex hull of the inner polytope (Figure 5.17c) that resulted from Algorithm G. The second row exhibits the similar results from the upside-down perspective.



(a) Online convex hull **(b)** Defect hull **(c)** Convex hull of defect hull

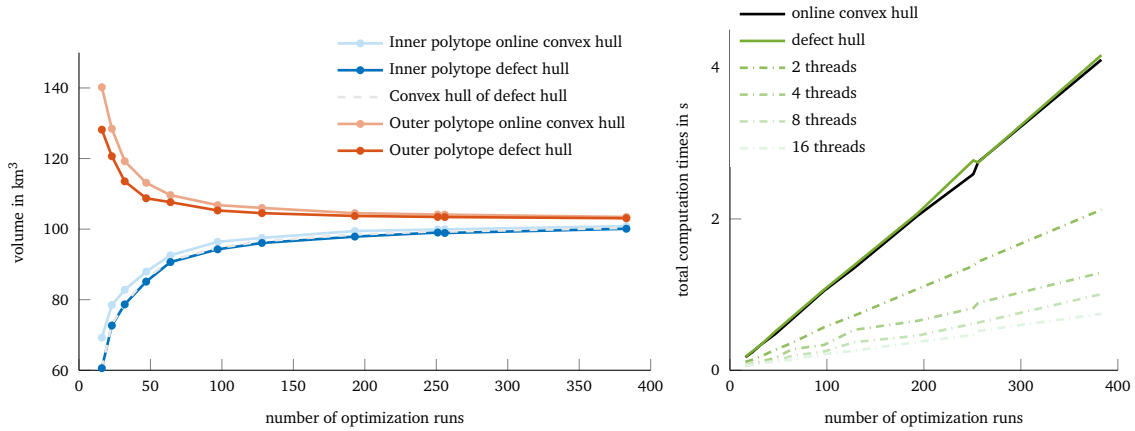
(d) Online convex hull **(e)** Defect hull **(f)** Convex hull of defect hull

Figure 5.17: Visualized approximations of the relaxed reachability set

Many analogies between the reachability and the controllability results can are observable here. The set has a slight tendency to positive p_y values caused by the positive initial velocity in p_y-direction in (5.30). Compared to the defect hull, the outcome from the online convex hull algorithm has a less regular triangulation but features a more distinctive shape on the upper part of the set. The relaxed reachable set's upper end converges similar to a cone. This is visible in Figure 5.17a but is not reflected in Figure 5.17b. It becomes apparent after the evaluation of the convex hull though. The convex hull shows a very similar triangulation in the lower part of the defect hull approximation. The relaxed reachability set looks like a rotated version of the relaxed controllability set as there is a cone-shaped and a smooth round part. The smooth part of the relaxed controllability set is flatter, while the relaxed reachable set is more extensive.

In Figure 5.18a, the convergence of the volumes of the generated polytopes can be observed as the number of optimizations increases. The inner approximation of the online convex hull algorithm covers more volume than the defect hull. The inner polytope of the defect hull approach is practically congruent to the convex hull of its vertices, with more than 99% of the space covered. Conversely, the outer approximation of the defect hull is

closer to the actual sought set than the outer approximation of the online convex hull. In the run time comparison in Figure 5.18b, the computation runs through slightly faster using the online convex hull algorithm in most configurations concerning the number of optimizations. It takes roughly the same time (2.75 s) to obtain the polytopes in Figures 5.17a and 5.17b. However, in the case of the defect hull approach, the computation time can be reduced to 1.3 s, 0.79 s, 0.58 s and 0.47 s as the number of threads doubles. The advantages due to the option of parallelization in Algorithm G are thus also evident here.

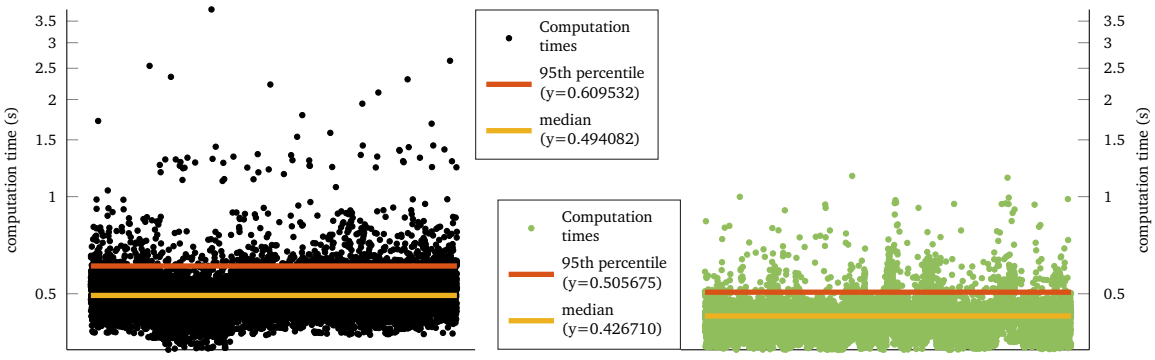


(a) Course of volumes

(b) Total computation times

Figure 5.18: Summarized results of the reachability scenario

It is checked in the same manner as in Section 5.2.3.1 whether the relaxed and the actual reachable sets coincide. Over 15,000 random elements in the defect hull's inner polytope approximating the relaxed reachable set are chosen. For each random element $\hat{z}$, grid task G($\hat{z}$) is solved subject to the nonlinear constraints (5.2), (5.13) and boundary values (5.30), (5.31). The respective optimal objective values practically vanish as well in this scenario. Hence, another indicator is found supporting the conjecture in [28] that the relaxed and the original reachability sets are equal.

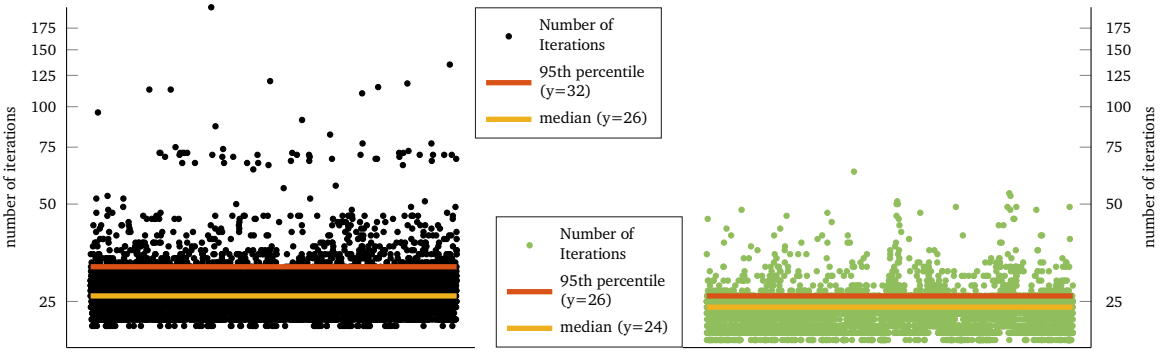


(a) Consistent initial guess

(b) Adaptive initial guess

Figure 5.19: Computation times using different initial guess strategy in the reachability scenario

A fast recalculation of the reference trajectory becomes necessary in a landing scenario if the landing site must be changed due to hazards or other harsh conditions at the original target position. For this purpose, the relaxed reachability set can be exploited as a library for initial guesses. Figure 5.19 compares the computation times between the consistent and the adaptive initial guess strategy. The median lies at 0.49 s in Figure 5.19a and can be reduced to 0.43 s in Figure 5.19b. The relative difference is 13.7%. The 95th percentile, originally at 0.61 s, is decreased to 0.51 s, which makes a relative difference of about 17%.



(a) Consistent initial guess **(b)** Adaptive initial guess

Figure 5.20: Number of iterations using different initial guess strategy in the reachability scenario

The improvements are visible in Figure 5.20 as well. The median for the necessary number of iterations is originally 26 and can be reduced to 24. The 95th percentile falls from 32 to 26 iterations. Relative differences of about 7.7% and 18.75% are therefore possible.
The relaxed reachability set states not only all feasible landing sites but also provides the means to find the corresponding trajectories and controls faster. Providing an adaptive initial guess benefits the solving of an NLP in terms of speed and robustness.

Chapter 6

Conclusion

This chapter concludes the present work. It summarizes the most important results and addresses possible starting points for further research.

6.1 Summary

This work has presented algorithms aiming to approximate reachability sets based on optimization techniques. The defect hull algorithm has been newly developed. The general approach of approximating sets by polytopes is not new, but the theoretical and algorithmic tricks to obtain the highest possible yield from a costly optimization. Theoretical foundations regarding geometry and numerical optimization were established for this and other presented algorithms. Essential elements were the parametric sensitivity analysis and convexity in the geometric and function-theoretical sense. With the help of parametric sensitivity analysis, both slope and curvature at boundary points of a set can be determined under certain conditions. Making this data accessible allows the interpolation of the boundary of the sets with only a few optimizations. The reconstruction by interpolation reduces the need for large numbers of optimization.
Convex optimization is a mature mathematical technology that has gained popularity in the aerospace domain. The class of SOCP generalizes linear programs without losing convergence properties. SOCP solvers reliably provide fast global solutions, essential in real-time applications. While convex optimization covers only a small fraction of all conceivable problem formulations, it is possible to determine the optimal control for a complex nonlinear application such as landing on the surface of Mars by formulating and solving a SOCP.
The grid, online convex hull, and defect hull algorithm have been presented. Every approach has its merits and produces an over-approximation of the set in the case of a convex set. The grid algorithm can cover most applications because it can handle sets of any shape. The online convex hull and defect hull algorithms produce polytopes and work best when the desired convex set. In the online convex hull algorithm, optimizations are performed sequentially. The result is a convex polytope. The defect hull algorithm allows the parallelization of optimizations. The resulting polytope may be non-convex, but it loses no

volume in the two-dimensional and only a tiny fraction of the volume of the actual convex hull in the three-dimensional. If the boundary is smooth, the defect hull algorithm, unlike the other approaches, can be used to determine the curvature pointwise.
As applications, two landing scenarios have been regarded. In the lunar lander model, coordinates are transformed into the dhc-framework. The resulting optimization problem is an NLP. In the Mars lander scenario, the dynamics and constraints are relaxed such that the feasibility problem is convex. The lossless convexification result has been presented. It states that the result of the relaxed feasibility problem adheres to the original dynamic and constraints for a specific objective. It is not proven that the reachability or controllability set of the relaxed feasibility problem can be equated with the corresponding set concerning the original constraints. However, there are hints that they are congruent. Nonetheless, the relaxed set provides an excellent basis to guarantee initial guesses with which nonlinear programs can be solved. The state trajectories and controls found with an SOCP solver that lead to boundary points of the relaxed dynamically constrained set were given to an NLP solver that searches for corresponding elements in the original feasibility set. The objective function of the nonlinear program was to get as close as possible to the boundary point.

6.2 Outlook

This work is a highly theoretical interpretation of how reachability analysis can be used in space missions. Many analyses would have to be run in advance for the actual online application of the defect hull algorithm during a critical mission phase. It must be determined how many optimizations can be allowed and are necessary. Accordingly, search directions would have to be determined. In order to exploit the full potential, parallel computing would have to be possible for the on-board computer. For embedded systems, due to the predefined number of optimization and the fixed number of facets, memory allocation can be done precisely without redundancy. In combination with convex optimization, the best conditions for an actual application exist here.
In the context of the landing scenario, the next step would be to overlay a topographic map, a hazard map, and the reachability set. Once a decision has been made on which landing site to select, a landing trajectory must be calculated. Since an excellent initial guess exists at this point, this trajectory could be determined through a nonlinear program. Here, however, excessive tests would have to be carried out in order to check whether a convergence of the NLP solver can be consistently brought about.
From a theoretical point of view, it would be interesting to know whether the dual variables can be specified in addition to the primary ones for the initial guesses. In SOCP solver, the number of Lagrange multipliers for a second-order cone constraint corresponds to the dimension of the cone, while in the NLP context, only one is necessary. A reasonable transformation of SOCP multipliers into NLP multipliers could benefit convergence. In case of relaxation, the back substitution would also have to be considered for the dual variables. The parametric sensitivity analysis for conic programs is another fascinating topic worth investigating to extend the theoretical foundation of the algorithms in this work. See [1] as a good starting point for further research.

The algorithms would theoretically also work in higher dimensions. However, a significantly larger number of optimizations are likely to be required to approximate a set of dimensions beyond three with sufficient accuracy. A major advantage of the defect hull algorithm is that the considered use cases in two- and three-dimensional space leads to a polytope that has no or only a slight deviation from the actual convex hull. Whether this condition also holds in higher dimensions needs verification. A helpful addition would be a parallel thread in which the convex hull is calculated and updated as soon as an optimization run is finished and a new vertex is available. If states of different units are considered in the dynamically constrained set, a suitable scaling approach would also have to be developed.
Great potential lies in the interpolation strategy based on the outcome of the defect hull algorithm. The presented one applies the Taylor expansion with an operating point closest to the evaluation point. Other strategies should be investigated as well. In particular, the case should be considered where the element to which a Taylor expansion maps may be outside the outer approximation. Either this element must be filtered in the course of this or re-evaluated on the basis of another working point. Overall, parametric sensitivity analysis provides a powerful tool to constrain the highest cost of optimization-based algorithms for computing reachability quantities: the optimization runs themselves.

Bibliography

[1] A. Agrawal, S. Barratt, S. Boyd, E. Busseti, and W. M. Moursi. Differentiating through a cone program. *Journal of Applied and Numerical Optimization*, 2019. doi:10.23952/jano.1.2019.2.02.

[2] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, 2003. doi:10.1007/s10107-002-0339-5.

[3] M. Althoff. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. PhD thesis, Technische Universität München, 2010.

[4] M. Althoff, C. Le Guernic, and B. H. Krogh. Reachable set computation for uncertain time-varying linear systems. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, HSCC '11, pages 93–102, Chicago, IL, USA, 2011. Association for Computing Machinery. doi:10.1145/1967701.1967717.

[5] Y. E. Arslantaş. *Development of a Reachability Analysis Algorithm for Space Applications*. PhD thesis, Universität Bremen, 2017.

[6] Y. E. Arslantaş, T. Oehlschlägel, and M. Sagliano. Safe landing area determination for a Moon lander by reachability analysis. *Acta Astronautica*, 128:607–615, 2016. doi:10.1016/j.actaastro.2016.08.013.

[7] A. V. Arutyunov and V. Obukhovskii. *Convex and Set-Valued Analysis: Selected Topics*. De Gruyter, 2016. doi:10.1515/9783110460308.

[8] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete & Computational Geometry*, 8(3):295–313, 1992. doi:10.1007/BF02293050.

[9] B. Açıkmeşe and S. R. Ploen. Convex Programming Approach to Powered Descent Guidance for Mars Landing. *Journal of Guidance, Control, and Dynamics*, 30(5):1353–1366, 2007. doi:10.2514/1.27553.

[10] R. Baier, C. Büskens, I. A. Chahma, and M. Gerdts. Approximation of reachable sets by direct solution methods for optimal control problems. *Optimization Methods and Software*, 22(3):433–452, 2007. doi:10.1080/10556780600604999.

[11] R. Baier. Mengenwertige Integration und die diskrete Approximation erreichbarer Mengen. 1995.

[12] R. Baier and M. Gerdts. A computational method for non-convex reachable sets using optimal control. In *2009 European Control Conference (ECC)*, pages 97–102, 2009. doi:10.23919/ECC.2009.7074386.

[13] S. Bansal and C. Tomlin. DeepReach: A Deep Learning Approach to High-Dimensional Reachability. *arXiv:2011.02082*, 2020.

[14] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hulls. *Acm Transactions on Mathematical Software*, 22(4):469–483, 1996.

[15] J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming*. Advances in design and control. Society for Industrial and Applied Mathematics, Philadelphia, 2nd edition, 2010. OCLC: ocn436309922.

[16] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, UK; New York, 2004. OCLC: 919495857.

[17] I. N. Bronštejn and K. A. Semendjaev, editors. *Taschenbuch der Mathematik*. Deutsch, Thun Frankfurt am Main, 5th edition, 2001.

[18] A. E. Bryson and Y.-C. Ho. *Applied optimal control: optimization, estimation, and control*. Hemisphere Pub. Corp. ; distributed by Halsted Press, Washington : New York, revised edition, 1975.

[19] C. Büskens. *Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustandsbeschränkungen*. PhD thesis, Westfälische Wilhelms-Universität Münster, 1998.

[20] 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, New York, NY, 2012. doi:10.1007/978-1-4614-4469-5_4.

[21] J. M. Carson, B. Açıkmeşe, and L. Blackmore. Lossless convexification of Powered-Descent Guidance with non-convex thrust bound and pointing constraints. In *Proceedings of the 2011 American Control Conference*, pages 2651–2656, San Francisco, CA, 2011. IEEE. doi:10.1109/ACC.2011.5990959.

[22] A. Cellina. A view on differential inclusions. *Rendiconti del Seminario Matematico*, 3, 2005.

[23] J. Ding and A. Zhou. Eigenvalues of rank-one updated matrices with some applications. *Applied Mathematics Letters*, 20(12):1223–1226, 2007. doi:10.1016/j.aml.2006.11.016.

[24] A. Domahidi. *Methods and tools for embedded optimization and control*. PhD thesis, ETH Zurich, 2013.

[25] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076, Zurich, 2013. IEEE. doi:10.23919/ECC.2013.6669541.

[26] D. Dueri, B. Açıkmeşe, M. Baldwin, and R. S. Erwin. Finite-horizon controllability and reachability for deterministic and stochastic linear control systems with convex constraints. In *2014 American Control Conference*, pages 5016–5023, 2014. doi:10.1109/ACC.2014.6859302. ISSN: 2378-5861.

[27] D. Dueri, S. V. Raković, and B. Açıkmeşe. Consistently improving approximations for constrained controllability and reachability. In *2016 European Control Conference (ECC)*, pages 1623–1629, 2016. doi:10.1109/ECC.2016.7810523.

[28] U. Eren, D. Dueri, and B. Açıkmeşe. Constrained Reachability and Controllability Sets for Planetary Precision Landing via Convex Optimization. *Journal of Guidance, Control, and Dynamics*, 38(11):2067–2083, 2015. doi:10.2514/1.G000882.

[29] J. Faraut and A. Koraányi. *Analysis on symmetric cones*. Oxford mathematical monographs. Clarendon Press ; Oxford University Press, Oxford : New York, 1994.

[30] A. V. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming*. Academic Press, New York, 1983. OCLC: 837975255.

[31] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. SIAM, 1990. Google-Books-ID: icWjpwgigkAC.

[32] O. Forster. *Analysis 2: Differentialrechnung im $\mathbb{R}^n$, gewöhnliche Differentialgleichungen*. Springer Fachmedien Wiesbaden, Wiesbaden, 2013. doi:10.1007/978-3-658-02357-7.

[33] C. Geiger and C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben: mit 140 Übungsaufgaben*. Springer, Berlin Heidelberg, 2002.

[34] A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In J. P. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, pages 257–271, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[35] H. H. Goldstine. *A History of the Calculus of Variations from the 17th through the 19th Century*, volume 5 of *Studies in the History of Mathematics and Physical Sciences*. Springer New York, New York, NY, 1980. doi:10.1007/978-1-4613-8106-8.

[36] J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012. doi:10.1016/j.ejor.2011.09.017.

[37] P. M. Gruber. *Convex and discrete geometry*. Number v. 336 in Grundlehren der mathematischen Wissenschaften. Springer, Berlin ; New York, 2007. OCLC: ocn123375612.

[38] B. Grünbaum. *Convex Polytopes*, volume 221 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 2003. doi:10.1007/978-1-4613-0019-9.

[39] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. doi:10.1007/BF02579150.

[40] W. Karush. *Minima of Functions of Several Variables with Inequalities as Side Conditions*. MSc Thesis, Department of Mathematics, University of Chicago, 1939.

[41] M. Knauer and C. Büskens. Real-Time Optimal Control Using TransWORHP and WORHP Zen. In G. Fasano and J. D. Pintér, editors, *Modeling and Optimization in Space Engineering*, volume 144, pages 211–232. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-10501-3_9.

[42] R. Kuhlmann. *A Primal-Dual Augmented Lagrangian Penalty-Interior-Point Algorithm for Nonlinear Programming*. PhD thesis, Universität Bremen, 2018.

[43] R. Kuhlmann, S. Geffken, and C. Büskens. WORHP Zen: Parametric Sensitivity Analysis for the Nonlinear Programming Solver WORHP. In N. Kliewer, J. F. Ehmke, and R. Borndörfer, editors, *Operations Research Proceedings 2017*, pages 649–654. Springer International Publishing, 2018. doi:10.1007/978-3-319-89920-6_86.

[44] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492. University of California Press, Berkeley and Los Angeles, Calif., 1951.

[45] O. Kunc and F. Fritzen. Generation of energy-minimizing point sets on spheres and their application in mesh-free interpolation and differentiation. *Advances in Computational Mathematics*, 45(5-6):3021–3056, 2019. doi:10.1007/s10444-019-09726-5.

[46] A. Kurzhanski and P. Varaiya. On Ellipsoidal Techniques for Reachability Analysis. Part I: External Approximations. *Optimization Methods and Software*, 17(2):177–206, 2002. doi:10.1080/1055678021000012426.

[47] C. Le Guernic. *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics*. PhD thesis, Université Joseph-Fourier - Grenoble I, 2009.

[48] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos. *Optimal control*. Wiley, Hoboken, 3rd edition, 2012.

[49] D. G. Luenberger. *Optimization by vector space methods*. Series in decision and control. Wiley, New York, 1968.

[50] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe. Convex Optimization for Trajectory Generation. *arXiv:2106.09125*, 2021.

[51] S. Mehrotra. On the Implementation of a Primal-Dual Interior Point Method. *SIAM Journal on Optimization*, 2(4):575–601, 1992. doi:10.1137/0802028.

[52] I. M. Mitchell. *Application of Level Set Methods to Control and Reachability Problems in Continuous and Hybrid Systems*. PhD thesis, Stanford University, 2002.

[53] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994. doi:10.1137/1.9781611970791.

[54] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer series in operation research and financial engineering. Springer, New York, 2nd edition, 2006.

[55] J. J. O'Connor and E. F. Robertson. MS Windows NT Kernel Description, 2002.

[56] T. Oehlschlägel, S. Theil, H. Krüger, M. Knauer, J. Tietjen, and C. Büskens. Optimal Guidance and Control of Lunar Landers with Non-throttable Main Engine. In F. Holzapfel and S. Theil, editors, *Advances in Aerospace Guidance, Navigation and Control*, pages 451–463. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-19817-5_34.

[57] G. V. Parshikov and A. R. Matviychuk. On resource-efficient algorithm for nonlinear systems approximate reachability set construction. In *Application of Mathematics in Technical and Natural Sciences*, page 120007, Albena, Bulgaria, 2017. doi:10.1063/1.5007424.

[58] H. J. Pesch and M. Plail. The maximum principle of optimal control: A history of ingenious ideas and missed opportunities. *Control and Cybernetics*, 38:973–995, 2009.

[59] H. J. Pesch and M. Plail. The Cold War and the Maximum Principle of Optimal Control. *Documenta Mathematica*, pages 243–253, 2012.

[60] F. A. Potra and S. J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1-2):281–302, 2000. doi:10.1016/S0377-0427(00)00433-7.

[61] R. T. Rockafellar. *Convex Analysis:*. Princeton University Press, 1970. doi:10.1515/9781400873173.

[62] M. Sagliano, D. Seelbinder, and S. Theil. SPARTAN: Rapid Trajectory Analysis via Pseudospectral Methods. In *8th International Conference on Astrodynamics Tools and Techniques*, Virtual Event, 2021.

[63] R. Sargent. Optimal control. *Journal of Computational and Applied Mathematics*, 124 (1-2):361–371, 2000. doi:10.1016/S0377-0427(00)00418-0.

[64] P. Schnabel. *Computertechnik-Fibel: Grundlagen der Computertechnik, Prozessortechnik, Halbleiterspeicher, Schnittstellen, Datenspeicher und Komponenten*. Patrick Schnabel, Ludwigsburg, 5th edition, 2020.

[65] R. Schneider. *Convex Bodies: The Brunn-Minkowski Theory*. Cambridge University Press, Cambridge, 2nd edition, 2013. doi:10.1017/CBO9781139003858.

[66] D. Seelbinder. *On-board Trajectory Computation for Mars Atmospheric Entry Based on Parametric Sensitivity Analysis of Optimal Control Problems*. PhD thesis, Universität Bremen, 2017.

[67] C. Tóth, J. O'Rourke, and J. E. Goodman. *Handbook of discrete and computational geometry*. CRC Press, Boca Raton, 3rd edition, 2017.

[68] W. Walter. *Gewöhnliche Differentialgleichungen: eine Einführung*. Springer-Lehrbuch. Springer, Berlin Heidelberg New York, 7th, completely revised and expanded edition, 2000.

[69] M. Wright. The interior-point revolution in optimization: History, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society*, 42 (1):39–56, 2005. doi:10.1090/S0273-0979-04-01040-7.

[70] Y. Yang. A Facet Enumeration Algorithm for Convex Polytopes. *arXiv:1909.11843 [math]*, 2021. arXiv: 1909.11843 version: 2.

[71] Y. Ye, M. J. Todd, and S. Mizuno. An O($\sqrt{n}L$)-Iteration Homogeneous and Self-Dual Linear Programming Algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994. doi:10.1287/moor.19.1.53.

[72] G. M. Ziegler. *Lectures on polytopes*, volume 152 of *Graduate texts in mathematics*. Springer-Verlag, New York, 1995.