INAUGURAL–DISSERTATION

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich–Mathematischen Gesamtfakultät

der

RUPRECHT–KARLS–UNIVERSITÄT
HEIDELBERG

vorgelegt von

Dipl.-Math. Holger Diedam

aus Salzkotten

Tag der mündlichen Prüfung:

....

Global Optimal Control Using Direct Multiple Shooting

Gutachter: Professor Dr. Sebastian Sager

Zusammenfassung

Das Ziel dieser Dissertation ist die Entwicklung eines neuen Algorithmus zur effizienten Bestimmung des globalen Optimums von Optimalsteuerungsproblemen. Im Unterschied zu bisherigen Methoden basiert dieser hier vorgestellte Ansatz auf dem direkten Mehrzielverfahren zur Diskretisierung des Optimalsteuerungsproblems, was in einer signifikanten Steigerung der Effizienz resultiert. Zur Relaxierung des diskretisierten Optimalsteuerungsproblems wird das sogenannte α-Branch-and-Bound Verfahren in Kombination mit validierter Integration verwendet.
Zum direkten Vergleich der auf dem direkten Einzielverfahren beruhenden Relaxierungen mit dem auf der direkten Mehrzielmethode basierenden Algorithmus werden mehrere theoretische Resultate bewiesen, die die Basis für die Effizienzsteigerung der neuen Methodik bilden. Eine speziell angepasste Branching-Strategie sorgt außerdem dafür, dass die durch das Mehrzielverfahren zusätzlich eingeführten Variablen den entstehenden Branch-and-Bound Baum nicht vergrößern. Des Weiteren wird eine adaptive Skalierung der verbreiteten Gershgorin Methode zur Abschätzung der Eigenwerte von Intervallmatrizen vorgestellt, die optimierte Relaxierungen liefert und damit allgemein zur Verbesserung der α-Branch-and-Bound Relaxierungen sowohl im Einziel- und Mehrzielverfahren beiträgt, als auch auf entsprechende Relaxierungen für nicht dynamische nichtlineare Probleme übertragbar ist. Zur weiteren Verbesserung der Laufzeit werden in dieser Arbeit außerdem noch Vorschläge im Bezug auf die benötigten Intervallsensitivitäten zweiter Ordnnung gemacht und eine Heuristik vorgestellt, welche nur auf einem bestimmten Unterraum relaxiert.
Der neu entwickelte Algorithmus ist, ebenso wie die auf dem Einzielverfahren basierte Alternative für den direkten Vergleich, in einem neu entwickelten Softwarepaket mit dem Namen GloOptCon implementiert. Das neue Verfahren wird genutzt um für eine Reihe von Benchmarkproblemen aus der Literatur und für bisher im Rahmen der globalen Optimalsteuerung noch wenig betrachteten Anwendungen das globale Optimum zu bestimmen. Die zusätzlich untersuchten Probleme stellen neue Herausforderungen, sowohl im Sinne der Größe, als auch dadurch, dass eines dieser Probleme seinen Ursprung in den sogenannten gemischt ganzzahligen Optimalsteuerungsproblemen hat, da es eine ganzzahlige zeitabhängige Steuerungsvariable enthält. Der bereits theoretisch bewiesene Effizienzgewinn wird durch die numerischen Ergebnisse bestätigt. Verglichen mit dem bisherigem Ansatz aus der Literatur wird die Anzahl der benötigten Iterationen für typische Probleme mehr als halbiert, während die Rechenzeit sogar um fast eine Größenordnung reduziert werden konnte. Dies wiederum erlaubt die globale Lösung von deutlich größeren Optimalsteuerungsproblemen.

Abstract

The goal of this thesis is the development of a novel and efficient algorithm to determine the global optimum of an optimal control problem. In contrast to previous methods, the approach presented here is based on the direct multiple shooting method for discretizing the optimal control problem, which results in a significant increase of efficiency. To relax the discretized optimal control problems, the so-called α-branch-and-bound method in combination with validated integration is used.

For the direct comparison of the direct single-shooting-based relaxations with the direct multiple-shooting-based algorithm, several theoretical results are proven that build the basis for the efficiency increase of the novel method. A specialized branching strategy takes care that the additionally introduced variables due to the multiple shooting approach do not increase the size of the resulting branch-and-bound tree. An adaptive scaling technique of the commonly used Gershgorin method to estimate the eigenvalues of interval matrices leads to optimal relaxations and therefore leads to a general improvement of the α-branch-and-bound relaxations in a single shooting and a multiple shooting framework, as well as for the corresponding relaxations of non-dynamic nonlinear problems. To further improve the computational time, suggestions regarding the necessary second-order interval sensitivities are presented in this thesis, as well as a heuristic that relaxes on a subspace only.

The novel algorithm, as well as the single-shooting-based alternative for a direct comparison, are implemented in a newly developed software package called GloOptCon. The new method is used to globally solve both a number of benchmark problems from the literature, and so far in the context of global optimal control still little considered applications. The additional problems pose new challenges either due to their size or due to having its origin in mixed integer optimal control with an integer-valued time-dependent control variable. The theoretically proven increase of efficiency is validated by the numerical results. Compared to the previous approach from the literature, the number of iterations for typical problems is more than halved, meanwhile the computation time is reduced by almost an order of magnitude. This in turn allows the global solution of significantly larger optimal control problems.

Danksagung

Ich möchte mich ganz herzlich bei meinen Betreuern Professor Dr. Sebastian Sager und Professor Dr. Dr. h.c. mult. Hans Georg Bock bedanken, die mich während meiner Promotion fortwährend unterstützt und gefördert haben. Durch ihre Anregungen, die gemeinsamen Diskussionen, aber auch durch ihr Vertrauen in mich wurde diese Arbeit möglich.
Für viele interessante Gespräche, spannende wissenschaftliche Anregungen, ebenso wie für die vielen entstandenen Freundschaften gebührt mein besonderer Dank der gesamten Arbeitsgruppe *Mathematical and Computational Optimization* die, umgezogen nach Magdeburg, zur Gruppe *Mathematical Algorithmic Optimization* wurde und der Gruppe *Simulation and Optimization*. Danke für diese gemeinsame Zeit.
Des Weiteren möchte ich mich bei Professor Dr. Paul Barton, Professor Dr. Moritz Diehl, Dr. Tran Van Hoai, Professor Dr. Katja Mombaur, Dr. Simon Jäger, Professor Dr. Dr. h.c. mult. Willi Jäger, Dr. Johannes Schlöder und Dr. Pierre-Brice Wieber explizit bedanken. Sie haben meine akademische Laufbahn aus ganz unterschiedlichen Gründen, beispielsweise in gemeinsamen Veröffentlichungen, Projekten oder Auslandsaufenthalten, signifikant geprägt.
Mein Dank geht auch an die Fakultät für Mathematik und Informatik der *Ruprecht-Karls-Universität Heidelberg*, die mir diese Promotion ermöglicht. Für die finanzielle und wissenschaftliche Unterstützung während dieser Zeit möchte ich mich außerdem bei der *Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences*, dem *Interdisziplinären Zentrum für Wissenschaftliches Rechnen der Ruprecht-Karls-Universität Heidelberg* und der *Otto-von-Guericke-Universität Magdeburg* bedanken.
Mein ganz persönlicher Dank geht an meine Mutter Anneliese Diedam, meinen Vater Josef Diedam, meinen Bruder Thorsten Diedam und meine Großeltern Maria und Julius Foth, die mich in meinem Werdegang jederzeit bestmöglich unterstützt haben. Ganz besonders möchte ich Kathrin Hatz danken, die immer für mich da ist.

Contents

Chapter 1

Introduction

The optimization of dynamic processes plays a crucial role for a wide range of applications in fields such as biology, chemistry, economy, engineering, physics and many more. These problems based on a dynamic process that is described by nonlinear differential equations with potentially nonlinear objective functions and additional nonlinear constraints are called optimal control problems (OCPs).

Whereas the local optimization of such problems advances rapidly, including partial differential equations [Pot11], multiple levels [Hat14], delay differential equations [Len14], mixed-integer decisions [Sag06], real-time applications [Kir10], robust solutions [HFD11b] or experimental design [KPBS12], finding the global optimum for such problems is still a very challenging, yet very important task, as many applications inhibit different local minima with potentially large differences in the objective function values. Those local minima sometimes occur due to obvious reasons such as symmetries in the movement of a robotic arm, but often are not predictable at all, as the analytical solution of the underlying differential equations is not known and the nonconvexities leading to such local minima are hidden in those equations. A general overview on non-dynamic global optimization is given in [FG09], whereas [HC14] starts with an overview of different approaches to solve OCPs in the context of global optimization.

In this thesis, we present a method to obtain deterministic global solutions for OCPs by transferring the α-branch-and-bound (αBB) method [AMF95, AF96] to a direct multiple shooting framework [BP84]. In comparison to the direct single-shooting-based approach [EF00b, PA02], we prove that adding multiple shooting nodes improves the convex relaxations and therefore results in a significant increase of efficiency. This is validated by means of numerical results.

1.1 Results of this Thesis

We present a novel mathematical algorithm for finding the global optimum of OCPs. We prove theoretically that this direct multiple-shooting-based algorithm is indeed a viable strategy that improves the efficiency and therefore allows to solve larger problems globally. These results are validated by means of numerical examples from the literature as well as real-world problems. The following points highlight and summarize the main results of this thesis in more detail.

A Multiple-Shooting-Based Algorithm for Global Optimal Control

Based on previous approaches that combine the α-branch-and-bound algorithm with the direct single shooting approach, we derive a novel multiple-shooting-based algorithm for global optimal control.

Theoretical Comparison and Quality of the Convexifications

We show that the multiple-shooting-based algorithm for the global optimization of optimal control problems is indeed a viable choice by proving theoretical results that compare the algorithm with the single-shooting-based variant and show how additional multiple shooting nodes increase the quality of the convexifications.

Treatment of Additional Variables

We derive a specialized branching strategy for the direct multiple-shooting-based algorithm and prove that the additional multiple shooting variables do not increase the size of the branch-and-bound tree.

An Adaptively Scaled Gershgorin Method

We use the scaled Gershgorin method to under- and overestimate the eigenvalues of the interval sensitivities. For the scaling vector, we suggest a fast heuristic based upon the iterative scaling proposed in [Hla15]. This provides us with fast means to scale the convex and concave relaxations that occur in the relaxed matching conditions independently. Furthermore, we give a different proof that provides more insight into the improvement gained over classical choices of the scaling vector.

Suggestions for Fast Bounds on the Second-Order Sensitivities

We propose to obtain the necessary second-order sensitivites not by validated integration of the variational differential equation, but by modifying the validated integrator such that it uses the Taylor model directly to obtain bounds on the sensitivities. This may speed up the algorithm significantly as shown in a proof-of-concept implementation.

Heuristics for a Reduced Space Relaxation

Experience with the direct multiple shooting method as a lifting of OCPs gives rise to an idea for a heuristic that relaxes the matching conditions on a certain subspace only and proves to be quite viable by speeding up the optimization by several orders of magnitude while not converging to a wrong local minimum a single time in our numerical results.

Software Package GloOptCon

We implement the proposed algorithm and the single-shooting-based variant for a direct comparison in our novel software package GloOptCon in order to obtain the numerical results. Various test functions, interfaces to different integrators and nonlinear program solvers and various plotting capabilities provide useful information when dealing with the global optimization of optimal control problems.

Numerical Results

We present a number of numerical results obtained by using our software package GloOptCon. The applications include a number of test problems from the global optimal control literature and, in the context of global optimization, novel real-world problems that contain new challenges. We systematically provide numerical confirmations for the previously proven theoretical results.

1.2 Thesis Overview

This thesis consists of three parts. The first part, consisting of Chapters 2 and 3, gives a short overview of the mathematical background necessary for the following chapters. The second part, consisting of Chapter 4 and 5, presents our theoretical advances as highlighted above and introduces the implementation in our software package GloOptCon. The last part presents the numerical results.

After a general introduction, we start with a short overview of direct optimal control in Chapter 2, focusing on the direct single shooting and direct multiple shooting approach. The first approach is the current state of the art in global optimal control, whereas the latter approach will be the basis for our new algorithm.

Chapter 3 presents the necessary basics from general non-dynamic deterministic global optimization, such as interval arithmetic, spatial branch-and-bound and convex relaxations. A section about common methods to obtain bounds on the eigenvalues of interval matrices follows. These bounds are necessary for the so-called α-branch-and-bound method that we use for the convex relaxations and validated integration methods. The last section in this chapter deals with methods to generate validated state bounds for differential equations in the form of Taylor-expansion-based validated integrators, which is important for deterministic global optimal control.

The following Chapter 4 focuses on our theoretical results, starting with the general idea, giving proofs for comparing results obtained with the direct single shooting approach and our final novel algorithm. We continue with sections on further improvements regarding an adaptively scaled Gershgorin method, an idea on faster bounds on the second-order sensitivities that are necessary for our relaxations and a heuristic that performs the relaxations on a reduced space. We close with notes on the global solution of global mixed integer optimal control problems.

Chapter 5 gives some insight in the implementation of our algorithm in the software package GloOptCon.

The last part, consisting of Chapter 6, applies our new method to a number of test problems from the literature and compares the results with our theoretical predictions of the performance of the direct single shooting approach versus the direct multiple-shooting-based global optimal control. We continue with an even larger optimal control application and one mixed integer optimal control problem, where multiple local minima are observed.

Part I

Introduction to Optimal Control and Global Optimization

Chapter 2

Direct Optimal Control

Optimal control is the optimization of an objective function with underlying nonlinear differential equations including time-dependent controls that govern the modeled process that is described by so-called state variables. In general, the controls are the unknowns in this problem class. In this thesis, we restrict ourselves to the case of ordinary differential equations (ODEs). This chapter gives a brief overview over the literature about solving such optimal control problems (OCPs) and introduces the problem formulation and notation that is needed in the following chapters.

Using the methods presented in this chapter, we are able to solve a large number of problems locally, but our test cases and applications solved globally in Chapter 6 give a first insight that even small OCPs often inhibit many local solutions. Some of these local solutions are obvious due to symmetries, for example during a robot arm movement on two symmetric paths, but many are not and the problem properties, such as nonconvexities and nonlinearities, are further concealed by the underlying ODEs that have to be solved numerically. Therefore, it is a very important task to determine global optimal solutions for this problem class. The main challenges arise from two sources. First, we have to determine bounds of the states that are governed by the underlying ODEs. Second, the time dependent controls naturally lead to very high dimensional and thus computationally expensive problems. A recent paper by Houska et al. [HC14] gives a short overview of different approaches to solve OCPs in the context of global optimization. It is important to note that most optimal control literature considers only underlying initial values problems (IVPs) with fixed or parametrized initial values only, whereas two of our applications are based on boundary value problems (BVPs) that contain fixed end points as well. An early method for the global solutions of such problems is given in [Loh92].

There are several methods to solve OCPs numerically. The first class is based on the Hamilton-Jacobi-Bellman (HJB) equation [Bel54, Bel57] and is called dynamic programming (DP). Using a cost-to-go function to calculate optimal arcs, this approach leads to a full tabulation of the state variables. Although DP is capable of identifying globally optimal solutions in a deterministic way, the tabulation limits the application to problems with a small number of state variables. In contrast, the so-called indirect or "first optimize, then discretize" approaches based on Pontryagin's Maximum Principle (PMP) [PBGM62, Boc81] formulate necessary optimality conditions of the OCP in function space resulting in a BVP that can be discretized and solved numerically. As all local optimality principles, PMP leads to locally optimal solutions and has not been extended to global optimal control so far.

In this thesis, we restrict ourselves to the so-called direct or "first discretize, then optimize" methods that discretize the underlying differential equations first and lead to a finite dimensional nonlinear problem. One method in this class, collocation [RS72, Bär83, Bie84], parametrizes both states and controls. Collocation is used for example in [EF00b] for global optimal control. Although the resulting NLPs are sparse, the high dimension makes solving them to global optimality hard [FTTMB08]. Instead, we focus on the direct single shooting [HR71, SS78] and direct multiple shooting [Pli81, BP84] method.

After presenting the general problem formulation considered in this thesis in the following section, we introduce both the direct single and the direct multiple shooting method. Direct single shooting is presented since it is the currently the most widely used approach in global optimal control, compare [FG09] for an extensive overview, and multiple shooting will be used in our new algorithm derived in Chapter 4. Since our optimization algorithms, as well as our convex relaxations method introduced in Chapter 3 are derivative-based, we dedicate the section afterwards to the derivatives of the solution trajectories, the so-called sensitivities.

2.1 Problem Formulation

For the remainder of this thesis, we consider a general OCPs of the following form

$$\begin{aligned}
\min_{u,p} \quad & \Phi_{\mathrm{M}}(x(t_N),p) + \int_{t_0}^{t_N} \Phi_{\mathrm{L}}(t,x(t),u(t),p)\,\mathrm{dt} && && \text{(2.1a)}\\
\text{s.t.} \quad & \dot{x}(t) = f(t,x(t),u(t),p) && \forall t \in [t_0,t_N] && \text{(2.1b)}\\
& 0 \le c(t,x(t),u(t),p) && \forall t \in [t_0,t_N] && \text{(2.1c)}\\
& 0 = r^{\mathrm{eq}}(x(t_0),\dots,x(t_N),p) && && \text{(2.1d)}\\
& 0 \le r^{\mathrm{ieq}}(x(t_0),\dots,x(t_N),p)\,. && && \text{(2.1e)}
\end{aligned}$$

The ODE describes the dynamic system between an initial time t_0 and a final time t_N. The time-dependent states $x : [t_0,t_N] \to \mathbb{R}^{n_x}$ are governed by the right-hand side (RHS) function $f : [t_0,t_N] \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$. The RHS depends on the time t, the states $x(t)$, a measurable control function $u : [t_0,t_N] \to \mathbb{R}^{n_u}$ and time-independent control values $p \in \mathbb{R}^{n_p}$ such as free components of the initial value $x(t_0)$ or even a free final time t_N. A given initial value, as well as more general constraints on interior or boundary values can be specified using the point-wise equality, respectively inequality constraints r^{eq} and r^{ieq}. Furthermore, general mixed control-state constraints $c(t,x(t),u(t),p)$ may be included. For existence and uniqueness of the ODE solution, we assume f to be locally Lipschitz continuous. The validated integrator methods we apply later are based on high-order Taylor expansions and therefore require that f is sufficiently often differentiable with respect to the time t. Furthermore, the nonlinear program (NLP) solvers and convexification used in this thesis assume that f, as well as c, r^{eq} and r^{ieq} are at least twice continuously differentiable.

The objective function of Bolza-type, consisting of a Mayer term $\Phi_{\mathrm{M}}(x(t_N),p)$ evaluated at the final time t_N only and a Lagrange term $\int_{t_0}^{t_N} \Phi_{\mathrm{L}}(t,x(t),u(t),p)\,\mathrm{dt}$ that is integrated over the time horizon, needs to be twice continuously differentiable. It is well known that these two types can be transformed into each other and therefore, we mainly focus on the Mayer term formulation to ease the notation. Details can be found for example in [Ces83].

2.2 Direct Methods for Optimal Control Problems

As mentioned in the introduction, there are many different methods to solve such OCPs numerically. In this next section, we focus on direct methods, namely the direct single and direct multiple shooting approach. We omit the widely used direct collocation method that approximates the control and state space using polynomial expressions. In contrast to that, direct single shooting relies on state-of-the-art numerical integrators to alter between simulating the ODE and iterating the optimization. Direct multiple shooting is a hybrid but nevertheless simultaneous approach, where the state trajectory is split into several independent parts by introducing new artificial initial values.

Direct Single Shooting

In direct single shooting, the control $u(t)$ in function space is approximated on a control discretization grid

$$t_0 = \tau_0 < \tau_1 < \cdots < \tau_M = t_N \,. \tag{2.2}$$

On each interval $[\tau_i, \tau_{i+1}], i \in \{0, \dots, M-1\}$ a local basis function $\tilde{u}_i(t, q_i)$ with a finite dimensional parametrization $q_i \in \mathbb{R}^{n_q}$ is used to approximate the true optimal control $u(t)$. Common choices for these functions are piecewise constant functions or piecewise linear approximations, the latter potentially with or without additional continuity conditions. Figure 2.1 shows an illustration of the direct single shooting method with different variants of the control discretization.

Combining all q_i into a single vector, we define the discretized control vector

$$q := \begin{pmatrix} q_0 \\ \vdots \\ q_{M-1} \end{pmatrix} \in \mathbb{R}^{Mn_q} \,. \tag{2.3}$$

An ODE solver is used to integrate from the potentially free initial value $x(t_0)$ to the final value $x(t_N)$ using the discretized controls $\tilde{u}_i(t, q_i)$. Suitable numerical integration methods range from classical explicit and implicit Runge-Kutta (RK) methods [Feh69] to backward differentiation formula (BDF) [CH52] implementations such as `DAESOL-II` [Alb10] or the `SUNDIALS` suite [HBG$^+$05a].

The mixed control-state constraints $c(t, x(t), u(t), p)$ are usually evaluated on a discretized grid as well. To ease the notation, we assume that this constraint grid coincides with the control discretization grid. Please note that this assumption is not necessary and our software package GloOptCon presented in Chapter 5 is not restricted in this regard. For further information on how to handle certain types of path constraints in detail we refer to [Pot06].

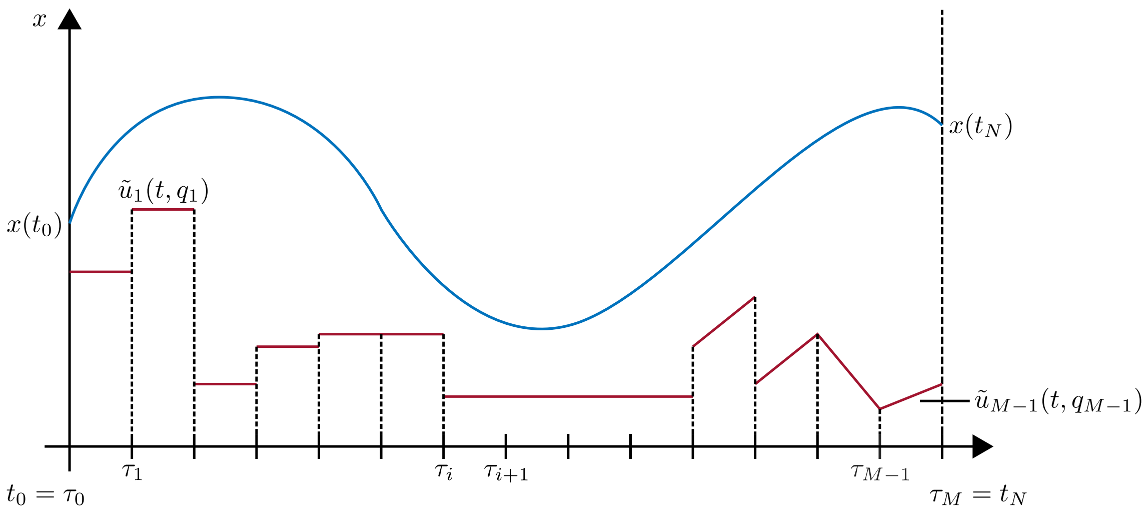


Figure 2.1: Visualization of the direct single shooting approach with state trajectory $x(t)$ between the initial value $x(t_0)$ and a final value $x(t_N)$ based upon the local control approximation $\tilde{u}_i(t, q_i)$ of different types defined on the control discretization grid.

Applying this control discretization to Problem (2.1) results in the discretized OCP

$$\begin{aligned}
\min_{q,p} \quad & \Phi_{\mathrm{M}}(x(t_N), p) + \sum_{i=0}^{M-1} \int_{t_i}^{t_{i+1}} \Phi_{\mathrm{L}}(t, x(t), \tilde{u}_i(t, q_i), p)\, \mathrm{d}t && \text{(2.4a)} \\
\text{s.t.} \quad & \dot{x}(t) = f(t, x(t), \tilde{u}_i(t, q_i), p) \qquad \forall t \in [t_i, t_{i+1}],\ \forall i \in \{0, \dots, M-1\} && \text{(2.4b)} \\
& 0 \le c(t, x(t), \tilde{u}_i(t, q_i), p) \qquad \forall t \in [t_i, t_{i+1}],\ \forall i \in \{0, \dots, M-1\} && \text{(2.4c)} \\
& 0 = r^{\mathrm{eq}}(x(t_0), \dots, x(t_N), p) && \text{(2.4d)} \\
& 0 \le r^{\mathrm{ieq}}(x(t_0), \dots, x(t_N), p)\,. && \text{(2.4e)}
\end{aligned}$$

To solve this discretized problem with an NLP solver iteratively, we have to alternate between simulations of the underlying ODE to obtain the states and optimization iterations of the, now finite dimensional, decision variables (q, p). In this sense the direct single shooting approach is a sequential method. The advantage, compared to collocation methods is that we can use state-of-the-art ODE solvers for the simulation. Furthermore, we have few degrees of freedom even for large ODE systems and we need only an initial guess for the discretized controls q and control values p. A first downside is that we are not able to include additional knowledge about the states $x(t)$ in the initial guess in the iterative method and the solution trajectory $x(t)$ may depend very nonlinearly on q. Therefore, unstable systems are often difficult to treat using the direct single shooting approach.
For theory and algorithms on how to solve the resulting NLPs, we refer to the textbook [NW06].

Direct Multiple Shooting

Direct multiple shooting introduces a second time grid, the multiple shooting nodes

$$t_0 < t_1 \cdots < t_N \tag{2.5}$$

with multiple shooting intervals $[t_i, t_{i+1}],\ i \in \{0, \dots, N-1\}$. Although this new grid may coincide with the control grid $\tau_0 < \tau_1 < \dots < \tau_M$, it is important to highlight that these two grids do not have to be equivalent. We present some numerical results in Chapter 6 using different grids.

For the sake of simplicity, Figure 2.2 illustrating the multiple shooting method shows only a single common grid. Furthermore, to ease the notation, we assume that the multiple shooting nodes are a subset of the control grid. This assumption allows us to merge the control parameterizations into a single vector q_i per multiple shooting node.

At each multiple shooting node t_i, we introduce a new auxiliary variable $s_i \in \mathbb{R}^{n_x},\ i \in \{0, \dots, N\}$. We refer to these vectors as multiple shooting variables. The multiple shooting variables $s_0, \dots, s_{N-1}$ act as new initial values for the corresponding integrations on each multiple shooting interval and at the last node t_n, s_n replaces the end value of the trajectory $x(t_N)$, for example in an occurring Mayer-type objective function. To ensure a continuous solution trajectory, we have to add so-called matching conditions to the problem. We define $x(t_{i+1}; s_i, q_i, p)$ as the solution trajectory at time t_{i+1} based upon initial values s_i at time t_i, the corresponding control discretization q_i in this multiple shooting interval and control values p. Please note that to ease the notation, the dependency of $x(t_{i+1}; s_i, q_i, p)$ on the local basis functions $\tilde{u}_i(t, q_i)$ is omitted. Using this definition, we can add equality constraints to enforce the end of the trajectory on an interval $[t_i, t_{i+1}]$ being equal to the next multiple shooting variable s_{i+1}:

$$x(t_{i+1}; t_i, s_i, q_i, p) - s_{i+1} = 0 \quad \forall i \in \{0, \dots, N-1\}\,. \tag{2.6}$$

These matching conditions ensure a continuous solution trajectory after convergence of the resulting discretized problem and are illustrated in Figure 2.3. Furthermore, we define the vector of all multiple shooting variables

$$s := \begin{pmatrix} s_0 \\ \vdots \\ s_N \end{pmatrix}. \tag{2.7}$$

Applying this discretization to Problem (2.1) results in the discretized OCP

$$\begin{aligned}
\min_{s,q,p} \quad & \Phi_{\mathrm{M}}(s_N, p) + \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \Phi_{\mathrm{L}}(t, x(t; t_i, s_i, q_i, p), \tilde{u}_i(t, q_i), p)\, \mathrm{d}t && \text{(2.8a)}\\
\text{s.t.} \quad & 0 = x(t_{i+1}; t_i, s_i, q_i, p) - s_{i+1} \qquad \forall i \in \{0, \dots, N-1\} && \text{(2.8b)}\\
& 0 \le c(t_i, s_i, \tilde{u}_i(t_i, q_i), p) \qquad \forall i \in \{0, \dots, N-1\} && \text{(2.8c)}\\
& 0 = r^{\mathrm{eq}}(s_0, \dots, s_N, p) && \text{(2.8d)}\\
& 0 \le r^{\mathrm{ieq}}(s_0, \dots, s_N, p)\,. && \text{(2.8e)}
\end{aligned}$$

Direct multiple shooting is in a sense hybrid between the direct collocation and direct single shooting. On the one hand, the resulting discretized problem is not as large as in the collocation case and can even be reduced to the size of the direct single shooting using a technique called condensing [BP84, Lei99]. On the other hand, we are still able to apply state-of-the-art ODE solvers as in the single shooting case. Nevertheless, and in contrast to direct single shooting, we are able to include additional information about the states $x(t)$ in the initial guesses

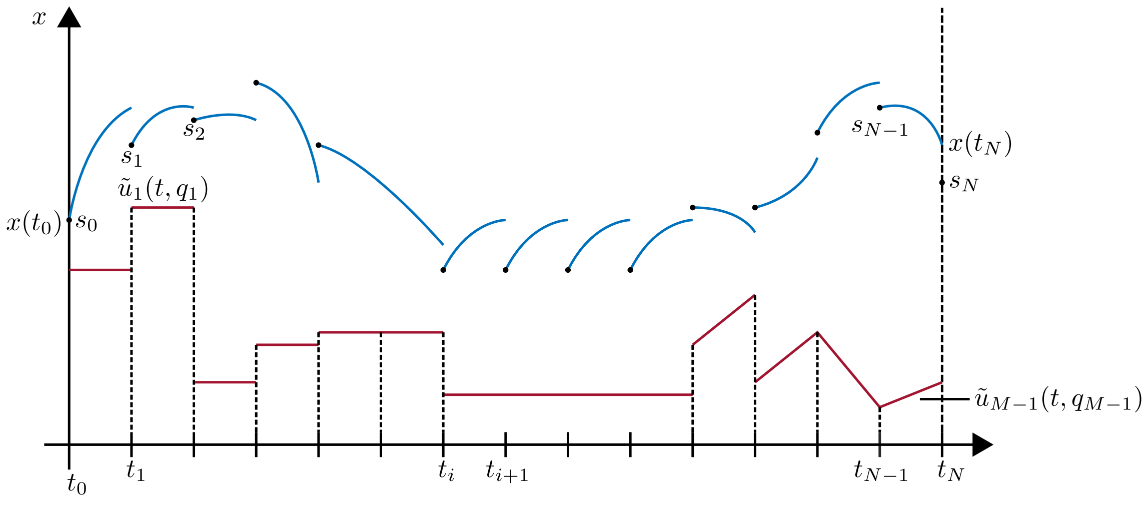


Figure 2.2: Visualization of the direct multiple shooting method with local state trajectories on the multiple shooting intervals $[t_i, t_{i+1}]$ based upon the corresponding initial values s_i and the local control approximation $\tilde{u}_i(t, q_i)$ of different type defined on the control discretization grid.

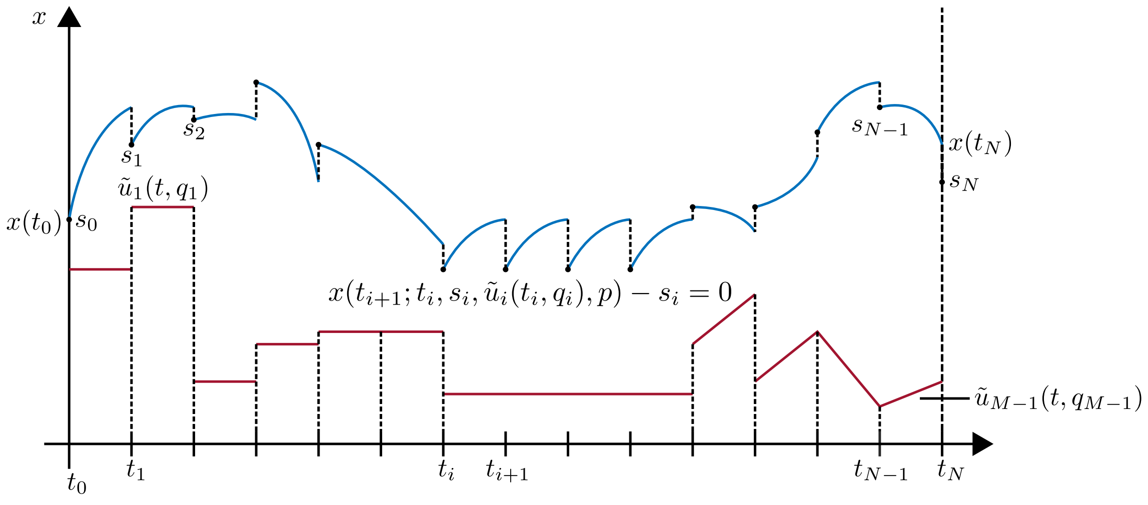


Figure 2.3: Visualization of the multiple shooting matching conditions in Equation (2.6) that ensure a continuous trajectory after convergence of the discretized Problem (2.8).

for the multiple shooting variables s_i. This results in an efficient method to solve OCPs that is able to solve unstable systems as well. For more details on multiple shooting, we refer to the in-depth introductions among others in [Lei99, Die02, Sag05, Kir10].

2.3 Sensitivities

Using a derivative-based NLP solver to solve the discretized Problem (2.4) or (2.8) requires the derivatives of the solution trajectory at some point $x(t)$, $t \in [t_0, t_N]$ with respect to the control parametrization variables q and control values p such as potentially free initial values $x(t_0)$ or a free final time t_N. Furthermore, the so-called sensitivities play a crucial role in the theoretical results in Chapter 4. We focus on the sensitivities necessary for the direct multiple shooting approach as the sensitivities for direct single shooting are a subset of these as indicated briefly at the end.

First-Order Sensitivities

Differentiating the trajectory $x(t;t_i,s_i,q_i,p)$ at a time t_{i+1} with respect to the initial values at the multiple shooting nodes t_i, the control discretization q_i, further free control values p and the time, we obtain the first-order sensitivities from the first-order variational differential equations [Boc87]. Using the unit matrix $I^n \in \mathbb{R}^{n\times n}$, the zero matrix $0^{n\times m} \in \mathbb{R}^{n\times m}$ and omitting the function arguments to ease the notation, we obtain on each multiple shooting interval $[t_i, t_{i+1}]$ the matrix-valued variational differential equations

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}x}{\mathrm{d}s_i} = \frac{\partial f}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s_i} + \frac{\partial f}{\partial s_i} = \frac{\partial f}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}s_i}, \qquad \frac{\mathrm{d}x(t_i)}{\mathrm{d}s_i} = I^{n_x} \qquad \forall i \in \{0,\dots,N-1\} \tag{2.9a}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}x}{\mathrm{d}q_i} = \frac{\partial f}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}q_i} + \frac{\partial f}{\partial \tilde{u}_i}\frac{\mathrm{d}\tilde{u}_i}{\mathrm{d}q_i}, \qquad \frac{\mathrm{d}x(t_i)}{\mathrm{d}q_i} = 0^{n_x\times n_q} \qquad \forall i \in \{0,\dots,N-1\} \tag{2.9b}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}x}{\mathrm{d}p} = \frac{\partial f}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}p} + \frac{\partial f}{\partial p}, \qquad \frac{\mathrm{d}x(t_i)}{\mathrm{d}p} = 0^{n_x\times n_p} \qquad \forall i \in \{0,\dots,N-1\} \tag{2.9c}$$

that hold true componentwise. We are already using the simplification that the RHS is independent of the initial values s_i and therefore $\frac{\partial f}{\partial s_i} = 0$. In the single shooting case, we need the derivatives obtained from Equations (2.9b) and (2.9c) as well. The sensitivities with respect to the initial value are only necessary for each free initial value $p' \in \mathbb{R}^{n_{p'}}, n_{p'} \leq n_x$. To ease the notation, we assume that the last $n_{p'}$ states have a free initial value and we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}x}{\mathrm{d}p'} = \frac{\partial f}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}p'}, \qquad \frac{\mathrm{d}x(t_0)}{\mathrm{d}p'} = \begin{pmatrix} 0^{(n_x-n_{p'})\times n_{p'}} \\ I^{n_{p'}} \end{pmatrix} \tag{2.10}$$

instead of Equation (2.9a).

Second-Order Sensitivities

The convex relaxation that we use to obtain globally optimal solutions and that is introduced in Chapter 3 is based on second-order derivatives and in the case of OCPs this means that their second-order sensitivities are required.

Differentiating Equations (2.9) once more and using the same simplification for the derivative with respect to potentially free initial values leads to the second-order variational differential equations, for example derived in [VBCB99]. We use the notation from [OB05] with $\otimes$ being the Kronecker product and the second derivative tensors of the vector valued functions written in matrix form composed of the derivatives of each component of the RHS functions such as

$$\frac{\partial^2 f}{\partial x \partial q_i} := \begin{pmatrix} \frac{\partial^2 f_0}{\partial x \partial q_i} \\ \vdots \\ \frac{\partial^2 f_{n_x-1}}{\partial x \partial q_i} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 f_0}{\partial x_0 \partial q_{i,0}} & \cdots & \frac{\partial^2 f_0}{\partial x_{n_x-1} \partial q_{i,0}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f_0}{\partial x_0 \partial q_{i,n_q-1}} & \cdots & \frac{\partial^2 f_0}{\partial x_{n_x-1} \partial q_{i,n_q-1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f_{n_x-1}}{\partial x_0 \partial q_{i,0}} & \cdots & \frac{\partial^2 f_{n_x-1}}{\partial x_{n_x-1} \partial q_{i,0}} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f_{n_x-1}}{\partial x_0 \partial q_{i,n_q-1}} & \cdots & \frac{\partial^2 f_{n_x-1}}{\partial x_{n_x-1} \partial q_{i,n_q-1}} \end{pmatrix}, \tag{2.11}$$

with additional subscripts indicating the corresponding vector components.

Furthermore, as the second-order variational differential equations are coupled with respect to s_i, q_i and p. Therefore, we introduce the vector of local decision variables

$$\tilde{s}_i := \begin{pmatrix} s_i \\ q_i \\ p \end{pmatrix} \tag{2.12}$$

on each multiple shooting interval $[t_i, t_{i+1}]$, containing the initial values, parametrized controls and control values with influence on the local state $x(t; t_i, s_i, q_i, p_i) = x(t; t_i, \tilde{s})$ with dimension $n_{\tilde{s}} = n_x + n_q + n_p$.

Using $\tilde{s}$, we can define the matrix form of the second-order sensitivity tensor

$$\frac{\mathrm{d}^2 x}{\mathrm{d}\tilde{s}^2} := \begin{pmatrix} \frac{\mathrm{d}^2 x_0}{\mathrm{d}\tilde{s}^2} \\ \vdots \\ \frac{\mathrm{d}^2 x_{n_x-1}}{\mathrm{d}\tilde{s}^2} \end{pmatrix}. \tag{2.13}$$

Using this notation, we can state the second-order variational differential equation as

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}^2 x}{\mathrm{d}\tilde{s}^2} = &\left(\frac{\partial f}{\partial x} \otimes I^{n_{\tilde{s}}}\right)\frac{\mathrm{d}^2 x}{\mathrm{d}\tilde{s}^2} + \left(I^{n_x} \otimes \left(\frac{\mathrm{d}x}{\mathrm{d}\tilde{s}}\right)^T\right)\left(\frac{\partial^2 f}{\partial x^2}\frac{\mathrm{d}x}{\mathrm{d}\tilde{s}} + \frac{\partial^2 f}{\partial x \partial \tilde{s}}\right) \\ &+ \left(\frac{\partial^2 f}{\partial x \partial \tilde{s}}\frac{\mathrm{d}x}{\mathrm{d}\tilde{s}} + \frac{\partial^2 f}{\partial \tilde{s}^2}\right). \end{aligned} \tag{2.14}$$

Please note that for the corresponding components of $\tilde{s}$ the same simplifications such as $\frac{\partial f}{\partial s_i} = 0$ and further application of the chain rule $\frac{\partial f}{\partial q_i} = \frac{\partial f}{\partial \tilde{u}_i}\frac{\mathrm{d}\tilde{u}_i}{\mathrm{d}q_i}$ apply for the corresponding components as seen for the first-order sensitivities.

Furthermore, assuming given first-order sensitivities $\frac{\mathrm{d}x}{\mathrm{d}\tilde{s}}$, the second-order sensitivities decouple with respect to the state components and we can reformulate Equation (2.14) $\forall i \in$

$0, \dots, n_x - 1$ as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}^2 x_i}{\mathrm{d}\tilde{s}^2} = \left(\frac{\partial f}{\partial x} \otimes I^{n_{\tilde{s}}}\right)\frac{\mathrm{d}^2 x_i}{\mathrm{d}\tilde{s}^2} + \left(\left(\frac{\mathrm{d}x}{\mathrm{d}\tilde{s}}\right)^T \left| \begin{array}{c} I^{n_{\tilde{s}} \times n_{\tilde{s}}} \\ \hline 0^{n_x \times n_{\tilde{s}}} \end{array}\right.\right) \begin{pmatrix} \frac{\partial^2 f_i}{\partial x^2} & \frac{\partial^2 f_i}{\partial x \partial \tilde{s}} \\ \frac{\partial^2 f_i}{\partial x \partial \tilde{s}} & \frac{\partial^2 f_i}{\partial \tilde{s}^2} \end{pmatrix} \left(\begin{array}{c|c} \multicolumn{2}{c}{\frac{\mathrm{d}x}{\mathrm{d}\tilde{s}}} \\ \hline 0^{n_{\tilde{s}} \times n_x} & I^{n_{\tilde{s}} \times n_{\tilde{s}}} \end{array}\right). \quad (2.15)$$

This formulation gives a bit more insight into the convex relaxations of the ODE in the next chapters.

Chapter 3

Global Optimization

The field of global optimization aims at identifying the global optimum of a nonconvex problem. In [Neu04] Neumaier classifies global optimization into four categories, ranging from incomplete and only asymptotically complete methods that have no means to quantify the quality of the currently best solution, to complete and rigorous methods that lead to bounds on the global optimum, in the last category even accounting for any rounding errors.

The wide range of different stochastic global optimization methods such as simulated annealing (SA) [KGV83] and genetic algorithms (GA) [Hol73] are often easy to implement, but fall into the first two categories. Therefore, those approaches do not yield lower bounds on the optimal solution. In contrast, in this thesis, we focus on the latter two categories, because for many applications, such as [SBD$^+$11], it is crucial to have a real measure for the quality of the solution in the form of an interval, in which the global optimal function value is guaranteed to be.

The survey [FG09] gives a broad overview over the literature in global optimization in general. The basis for those methods and for most deterministic global optimization algorithms in general are interval arithmetics [Moo66] and a spatial branch-and-bound technique [LD60, LMSK63]. Both fundamentals are briefly introduced in this chapter.

More complex algorithms for deterministic global optimization rely on convex relaxations of the original problem. In this chapter, we focus on the α-branch-and-bound (αBB) method that is described for example in [AMF95, AF96, AAF98]. Apart from this particular convexification method, there are other convexification techniques applied successfully in global optimization and global optimal control in particular, such as McCormick relaxations [MCB09, SSB11] and polyhedral outer approximations [TS05].

The convex relaxations used in the αBB algorithm are based on the eigenvalues of interval Hessians that are the bounds on the second derivatives. The authors in [SW14] give an overview and comparison of different methods to determine bounds on those eigenvalues. We focus on a method based on Gershgorin's circle theorem suggested in [AAF98] and present this approach briefly in the last section of this chapter.

Finally, any deterministic global optimization approach for optimal control problems (OCPs) as introduced in Equation (2.1) relies on bounds on the states of the ordinary differential equations (ODEs). We obtain those bounds through so-called validated integration that aims at enclosing all possible solution trajectories based on an interval of feasible initial values, controls or control parameters. An early overview of different methods is given in [Rih94], whereas [Ned06] focuses on the different software implementations available. Section 3.6 is dedicated to obtaining bounds of the states of an OCP using those methods. These bounds in combination with a spatial branch-and-bound are sufficient for a simple single-shooting-based deterministic global optimization algorithm that is described in this section. Furthermore, using validated integrators allows us to obtain state bounds on the second-order variational differential equalities (2.14), resulting in bounds on the sensitivities that allow the application of the αBB algorithm to OCPs. We close the chapter by describing the single-shooting-based αBB algorithm for OCPs.

3.1 Constrained Nonlinear Optimization

In this first section, we restrict ourselves to the global optimization of nonlinear programs (NLPs), because as we have seen in the last chapter, our discretized optimal control problem (OCP) in Equation (2.8) is such an NLP. Only the last section on validated integrators is tailored to the special case of global optimal control. Furthermore, one general assumption is that all optimization variables have at least box constraints. Such constraints are usually given in most applications where, e.g., a controlled temperature is within specified limits, a motor has certain constraints on its torque and so on.
A standard constrained nonlinear program (NLP) has the following form

$$\begin{aligned} \min_{v} \quad & \phi(v) \\ \text{s.t.} \quad & g(v) = 0 \\ & h(v) \geq 0 \\ & v \in V \subset R^{n_v},\ V \text{ compact} \end{aligned} \tag{3.1}$$

with objective function $\phi : \mathbb{R}^{n_v} \to \mathbb{R}$, equality constraints $g : \mathbb{R}^{n_v} \to \mathbb{R}^{n_{\text{eq}}}$ and inequality constraints $h : \mathbb{R}^{n_v} \to \mathbb{R}^{n_{\text{ieq}}}$ that are all assumed to be sufficiently smooth. We directly recognize the discretized multiple shooting problem from Equation (2.8) as an NLP with a special structure. It is important to exploit this structure as we will see in Chapter 4. For now, we focus on such general NLPs to introduce the concepts used for global optimization. Finally, we note that an NLP is convex if and only if the objective function and the feasible set are convex.
The theory and algorithms to obtain local solutions of such NLPs can be found amongst others in [NW06].

3.2 Interval Arithmetic

Interval arithmetic or analysis is described in detail in the textbook [MKC09]. We summarize the mathematical notation that is used in the following sections and chapters.
We denote the closed interval Z indicated by an uppercase character as

$$Z := [\underline{z}, \overline{z}] = \{z \in \mathbb{R} : \underline{z} \leq z \leq \overline{z}\} \tag{3.2}$$

with lower bound $\underline{z} \in \mathbb{R}$ and upper bound $\overline{z} \in \mathbb{R}$, indicated by underlined and overlined variables. We denote the set of closed intervals with $[\mathbb{R}] := \{[\underline{z}, \overline{z}] : \underline{z} \leq \overline{z}, \underline{z} \in \mathbb{R}, \overline{z} \in \mathbb{R}\}$.
For basic operations $\odot$ on such intervals Z_1 and Z_2, we expect the resulting interval to contain all possibilities as follows

$$Z_1 \odot Z_2 = \{z_1 \odot z_2 : z_1 \in Z_1, z_2 \in X_2\}\ . \tag{3.3}$$

Using the so-called endpoint notation from Equation (3.2), set operations for addition, subtraction, multiplication and division that satisfy the requirement above are

$$Z_1 + Z_2 = \left[\underline{z}_1 + \underline{z}_2, \overline{z}_1 + \overline{z}_2\right] \tag{3.4a}$$
$$Z_1 - Z_2 = \left[\underline{z}_1 - \overline{z}_2, \overline{z}_1 - \underline{z}_2\right] \tag{3.4b}$$
$$Z_1 Z_2 = \left[\min\{\underline{z}_1\underline{z}_2, \underline{z}_1\overline{z}_2, \overline{z}_1\underline{z}_2, \overline{z}_1\overline{z}_2\}, \max\{\underline{z}_1\underline{z}_2, \underline{z}_1\overline{z}_2, \overline{z}_1\underline{z}_2, \overline{z}_1\overline{z}_2\}\right] \tag{3.4c}$$
$$\frac{Z_1}{Z_2} = Z_1 \frac{1}{Z_2} \quad \text{with} \quad \frac{1}{Z_2} = \{z_2 : \frac{1}{z_2} \in Z_1\} = \left[\frac{1}{\overline{z}_2}, \frac{1}{\underline{z}_2}\right]. \tag{3.4d}$$

Furthermore, we denote the width of an interval Z by $\mathrm{w}(Z) := \overline{z} - \underline{z}$, the midpoint by $\mathrm{m}(Z) := \frac{1}{2}(\underline{z} + \overline{z})$ and finally the absolute value by $|Z| := \max(|\underline{z}|, |\overline{z}|)$.
In this thesis, we usually deal with interval vectors, such as the box constrained domain of an NLP, or even interval matrices, such as the bounds on the Hessian of a function. In this case, all interval operations are meant to be componentwise. Let $\underline{v} \in \mathbb{R}^{n_v}$ and $\overline{v} \in \mathbb{R}^{n_v}$, then we define

$$V := \begin{pmatrix} [\underline{v}_0, \overline{v}_0] \\ \vdots \\ [\underline{v}_{n_v-1}, \overline{v}_{n_v-1}] \end{pmatrix} \tag{3.5}$$

with the corresponding set of all interval vectors $[\mathbb{R}^{n_v}]$ of dimension n_v.
The width, midpoint and absolute value of an interval vector are then denoted by

$$\mathrm{w}(V) := \begin{pmatrix} \mathrm{w}([\underline{v}_0, \overline{v}_0]) \\ \vdots \\ \mathrm{w}([\underline{v}_{n_v-1}, \overline{v}_{n_v-1}]) \end{pmatrix}, \tag{3.6}$$

$$\mathrm{m}(V) := \begin{pmatrix} \mathrm{m}([\underline{v}_0, \overline{v}_0]) \\ \vdots \\ \mathrm{m}([\underline{v}_{n_v-1}, \overline{v}_{n_v-1}]) \end{pmatrix} \text{ and} \tag{3.7}$$

$$|V| := \begin{pmatrix} |[\underline{v}_0, \overline{v}_0]| \\ \vdots \\ |[\underline{v}_{n_v-1}, \overline{v}_{n_v-1}]| \end{pmatrix}. \tag{3.8}$$

and accordingly for interval matrices.

3.3 Branch-and-Bound

Branch-and-bound (BB) methods are extensively described in [Sch11]. For an early application in global optimization, we refer to [FS69], whereas our notation is based on the version found in [HT96, TH88]. In [SKS13] the authors deal with the problems that stem from nonconvex constraints.
The general idea is that any feasible solution of an NLP denotes an upper bound $\overline{f}$ for the global solution $f(x^*_{\mathrm{glo}})$ and the local optimum of a convex relaxation NLP$^{\mathrm{cv}}$, which is equal to

the global optimum in this case, is a corresponding lower bound $\underline{f}$ of $f(x^*_{\text{glo}})$. To refine those bounds, we branch on the optimization variables x. This means, we split their domain into two or more disjoint sets and treat the resulting subproblems that have a smaller feasible set separately. Figure 3.1 illustrates the idea for a one dimensional unconstrained problem.

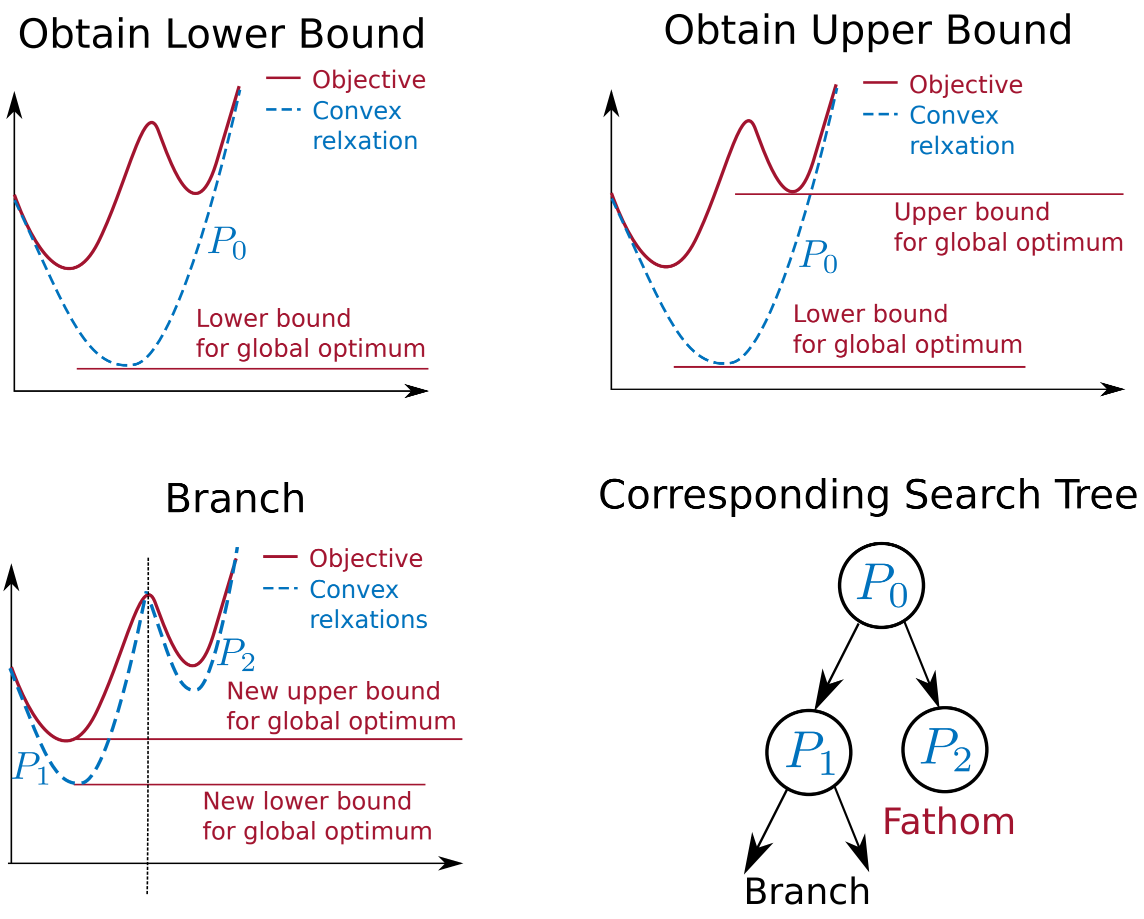

Obtain Lower Bound
Objective
Convex relxation
P_0
Lower bound for global optimum
Obtain Upper Bound
Objective
Convex relxation
Upper bound for global optimum
P_0
Lower bound for global optimum
Branch
Objective
Convex relxations
P_2
New upper bound for global optimum
P_1
New lower bound for global optimum
Corresponding Search Tree
P_0
P_1
P_2
Fathom
Branch

Figure 3.1: An illustration of the general branch-and-bound idea.

The resulting set of problems can be viewed as a tree structure, where we divide the domain further and further till we have achieved a desired interval width of $f(x^*_{\text{glo}}) \in [\underline{f}, \overline{f}]$ with $\text{w}([\underline{f}, \overline{f}]) \leq \epsilon$. In Algorithm 1, we formalize the spatial branch-and-bound approach.
To solve the resulting nonconvex and convex subproblems, different methods and solvers where mentioned in Section 3.1. Depending on the size of the problem and the memory necessary for warmstart information, a major speed up can usually be gained by saving additional data along with the nodes to improve the initial guess for the child nodes.
Two important customization decisions in the branch-and-bound algorithm are the problem and branching index selection. As all lower bounds must either be improved to convergence or till the corresponding node can be pruned, the former is usually done by selecting the node with the global bounds. The latter question is more complex and it is important to highlight that the selection of the next branching interval is not arbitrary, but has to follow certain rules to proof convergence. Furthermore, the selection method usually has a major influence on the convergence speed, compare for example [EF00a] or [PA02] for comparisons of αBB with

Algorithm 1: A basic spatial branch-and-bound algorithm for global optimization.

```
Let $\underline{\phi}$ and $\overline{\phi}$ denote the lower and upper bound on the global optimum of the
objective function of an NLP (3.1), $v \in V_0$ the initial compact domain of $v$ and $\epsilon$ a
desired accuracy for the global optimum.
**Solve** NLP locally on $V_0$ and obtain solution $v_0^*$ and objective value $\phi(v_0^*)$.
**Relax** NLP on $V_0$ and obtain convex problem $\mathrm{NLP}_0^{\mathrm{cv}}$.
**Solve** $\mathrm{NLP}_0^{\mathrm{cv}}$ locally (=globally) on $V_0$ and obtain solution $v_0^{\mathrm{cv},*}$ and objective value
$\phi_0^{\mathrm{cv}}(v_0^{\mathrm{cv},*})$.
**Initialize** a set of pairs of domains with corresponding local lower bounds
$\mathcal{V} := \{(V_0, \phi_0^{\mathrm{cv}}(v_0^{\mathrm{cv},*}))\}$.
**Initialize** bounds $\underline{\phi} := {\phi_0}^{\mathrm{cv}}(v_0^{\mathrm{cv},*})$, $\overline{\phi} := \phi(v^*)$, currently best solution $v^* := v_0^*$ and an
iteration counter $i := 0$ that serves as an index only.
**while** $\mathcal{V} \neq \emptyset$ **do**
    **if** $\overline{\phi} - \underline{\phi} \leq \epsilon$ **then**
        **Stop** algorithm, global solution obtained.
    **end**
    **Increment** iteration counter $i := i + 1$.
    **Select** pair $(V_i, \phi^{\mathrm{cv}}(v_i^{\mathrm{cv},*})) \in \mathcal{V}$.
    **Branch** domain $V_i$ into $N$ disjoint nonempty subdomains $W_{i,1}, \dots W_{i,N}$ such that
    $V = \bigcup_{k \in \{1,\dots,N\}} W_{i,k}$.
    **for** $k = 1$ **to** $N$ **do**
        **Solve** NLP locally on $W_{i,k}$ and obtain solution $w_{i,k}^*$ and objective value $\phi(w_{i,k}^*)$.
        **if** $\phi(w_{i,k}^*) < \overline{\phi}$ **then**
            **Update** upper bound $\overline{\phi} := \phi(w_{i,k}^*)$.
            **Update** best known solution $v^* := w_{i,k}^*$.
        **end**
        **Relax** NLP on $W_{i,k}$ and obtain convex problems $\mathrm{NLP}_{i,k}^{\mathrm{cv}}$.
        **Solve** $\mathrm{NLP}_{i,k}^{\mathrm{cv}}$ locally (=globally) on $W_{i,k}$ and obtain solution $w_{i,k}^{\mathrm{cv},*}$ and objective
        value $\phi_k^{\mathrm{cv}}(w_{i,k}^{\mathrm{cv},*})$.
    **end**
    **Set** $\mathcal{V} := \left(\mathcal{V} \setminus \left(V_i, \phi^{\mathrm{cv}}(v_i^{\mathrm{cv},*})\right)\right) \bigcup_{k \in \{1,\dots,N\}} \{(W_{i,k}, \phi_{i,k}^{\mathrm{cv}}(w_{i,k}^{\mathrm{cv},*}))\}$.
    **Set** new global lower bound $\underline{\phi} := \min\{\phi^{\mathrm{cv}}(v^{\mathrm{cv},*}) : (V, \phi^{\mathrm{cv}}(v^{\mathrm{cv},*})) \in \mathcal{V}\}$ .
    **Fathom:** remove all pairs with worse local lower bound than the global upper
    bound, i.e. $\mathcal{V} := \mathcal{V} \setminus \{\phi^{\mathrm{cv}}(v^{\mathrm{cv},*}) : (V, \phi^{\mathrm{cv}}(v^{\mathrm{cv},*})) \in \mathcal{V}, \overline{\phi} \leq \phi^{\mathrm{cv}}(v^*)\}$ .
**end**
The true global optimum is now bounded: $\phi(v_{\mathrm{glo}}^*) \in [\underline{\phi}, \overline{\phi}]$ with $\mathrm{w}([\underline{\phi}, \overline{\phi}]) \leq \epsilon$ .
Furthermore, the best solution found $v^*$ is either the global optimum or has an
objective function value within the desired accuracy.
```

different branching strategies on OCPs. A simple choice is to use the component with the largest interval width such that

$$\forall i, j \in \{1, \dots, n_x\} : \mathrm{w}([\underline{x}_i, \overline{x}_i]) \geq \mathrm{w}([\underline{x}_j, \overline{x}_j]) \, . \tag{3.9}$$

Scaling this, leads to the so called “least reduced axis method”, that is

$$\min_i \frac{\overline{x}_i - \underline{x}_i}{\overline{x}_{i,0} - \underline{x}_{i,0}} \, . \tag{3.10}$$

Convex-relaxation-specific versions exist and we present one tailored version for αBB in Section 3.4.
After choosing the index i, we have to branch into disjoint sets. For the implementation in Chapter 5, we choose the a bisection defined as follows

$$X_0 = \begin{pmatrix} [\underline{x}_1, \overline{x}_1] \\ \dots \\ [\underline{x}_i, \mathrm{m}(X_I)] \\ \dots \\ [\underline{x}_{n_x}, \overline{x}_{n_x}] \end{pmatrix}, X_1 = \begin{pmatrix} [\underline{x}_1, \overline{x}_1] \\ \dots \\ [\mathrm{m}([X_I]), \overline{x}_i] \\ \dots \\ [\underline{x}_{n_x}, \overline{x}_{n_x}] \end{pmatrix} . \tag{3.11}$$

Another crucial element in the BB algorithm is the convex relaxation of the NLPs on the current domain. In Section 2, we present methods to generate the necessary convex relaxations, focusing on the so-called αBB method for general nonconvex twice differentiable functions.

3.4 Convex Relaxations

Obtaining convex relaxations of a nonconvex problem can be achieved in many different ways. The global solvers BARON [TS05, Sah13] and Couenne [Bel09] decompose a nonconvex function into elementary functions and replace those with known convex underestimations. Another popular approach is to use McCormick relaxations [McC76] that were generalized later to not only create convex under- and overestimators of factorable functions, but even of whole algorithms [MCB09]. McCormick relaxations generate nonsmooth relaxations that in turn restrict the choice of an optimization algorithm especially in the presence of nonlinear constraints.
The basis for our results in the next chapters is the αBB relaxation that generates differentiable convex underestimations. The αBB algorithm uses second-order derivatives of a function to generate a convex underestimation and with an opposite sign in the following equations, a concave overestimation.
Whereas it is suggested in [AAF98] to use exact convex underestimations where possible, in general, any twice differentiable nonconvex function $f : \mathbb{R}^{n_x} \to \mathbb{R}$ may be convexified using a sufficiently large quadratic term [MF94]:

$$f^{\mathrm{cv}}(x) = f(x) + \sum_{i=1}^{n_x} \alpha_i (\overline{x}_i - x_i)(\underline{x}_i - x_i) \, ,$$

with $\overline{x}$ and $\underline{x}$ being the bounds of the domain of x and the components α_i of $\alpha \in \mathbb{R}^{n_x}$ sufficiently large. This is the so called nonuniform diagonal shift, as we use a specific α_i for each variables component. On the one hand, a simpler version is obtained using only a single α_0, the so-called uniform shift, whereas on the other hand, generalized versions [SWMF12] based on nondiagonal shifts exists.

To choose α sufficiently large, we check the convexity of a function f with the positive definiteness of the corresponding Hessian $\nabla^2 f$, see [BV04] amongst others:

Theorem 1 (Convexity of twice differentiable functions)**.** *Let $f : X \subseteq \mathbb{R}^{n_x} \to \mathbb{R}$ be twice differentiable. Then f is convex on X if and only if the domain X of f is convex and the Hessian is positive semidefinite, i.e. $\forall x \in X : \nabla^2 f(x) \succeq 0$.*

Applied to our convex relaxation f^{cv}, we obtain the following condition for positive definiteness:

$$\forall x \in X : \nabla^2 f^{\text{cv}}(x) + 2\text{diag}(\alpha) \succeq 0 \,. \tag{3.12}$$

Therefore, using the smallest eigenvalue $\lambda_{\min}(\nabla^2 f^{\text{cv}}(x), \forall x \in X)$ over the whole domain of x, we derive that a sufficiently large choice of α is

$$\alpha_i \geq \max(0, -\frac{1}{2}\lambda_{\min}(\nabla^2 f^{\text{cv}}(x), \forall x \in X)) \,. \tag{3.13}$$

The Hessian over the whole domain is an interval matrix. Therefore, we do not calculate this eigenvalue exactly, but under- and overestimate the eigenvalues using the methods described in the next section.

Furthermore, we are able to deduce an analogous concave overestimation estimation rule, demanding that the concave overstimation

$$f^{\text{cc}}(x) = f(x) + \sum_{i=1}^{n_x} \alpha_i^{\text{cc}}(\overline{x}_i - x_i)(\underline{x}_i - x_i) \tag{3.14}$$

of f is negative semidefinite:

$$\forall x \in X : \nabla^2 f^{\text{cc}}(x) + 2\text{diag}(\alpha^{\text{cc}}) \preceq 0 \,. \tag{3.15}$$

Once more, we deduce an α^{cc} that is sufficiently large if

$$\alpha_i \geq \min(0, -\frac{1}{2}\lambda_{\max}(\nabla^2 f^{\text{cc}}(x), \forall x \in X)) \,. \tag{3.16}$$

To get a better impression of this relaxation technique, Figure 3.2 shows a comparison of a manual, exact convex relaxation of the bilinear function $f(x) = x_1 x_2$ relaxed on the domain $X = ([-5,5],[-5,5])^T$. For bilinear terms the exact convex relaxation coincides with the following McCormick relaxation

$$\begin{aligned} f(x) &\geq \underline{x}_2\underline{x}_1 + \underline{x}_1 x_2 - \underline{x}_2\underline{x}_1 & \qquad f(x) &\leq \underline{x}_2 x_1 + \overline{x}_1 x_2 - \underline{x}_2\overline{x}_1 \\ f(x) &\geq \overline{x}_2\overline{x}_1 + \overline{x}_1 x_2 - \overline{x}_2\overline{x}_1 & \qquad f(x) &\leq \overline{x}_2 x_1 + \underline{x}_1 x_2 - \overline{x}_2\underline{x}_1 \,. \end{aligned}$$

The αBB relaxation is based on the corresponding Hessian

$$\nabla^2 f(x) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{3.18}$$

with eigenvalues $\lambda_1 = -1$ and $\lambda_2 = 1$. Therefore, according to Equation (3.13), choosing $\alpha = \frac{1}{2}$ is sufficiently large to guarantee convexity and we obtain

$$f^{\mathrm{cv}}(x) = x_1 x_2 + \frac{1}{2}\left((5 - x_0)(-5 - x_0) + (5 - x_1)(-5 - x_1)\right) . \tag{3.19}$$

We note that McCormick is a piecewise linear approximation, whereas the αBB uses a sufficiently large quadratic term to overpower any nonconvexities in the function. Please note that this single example gives by no means an insight into the approximation and convergence quality for more complex functions. We refer to [BM12] for a comparison of the convergence rates of these methods showing that despite αBB relaxations often being quite weak for large domains, the convergence when reducing the domain size is of quadratic order.

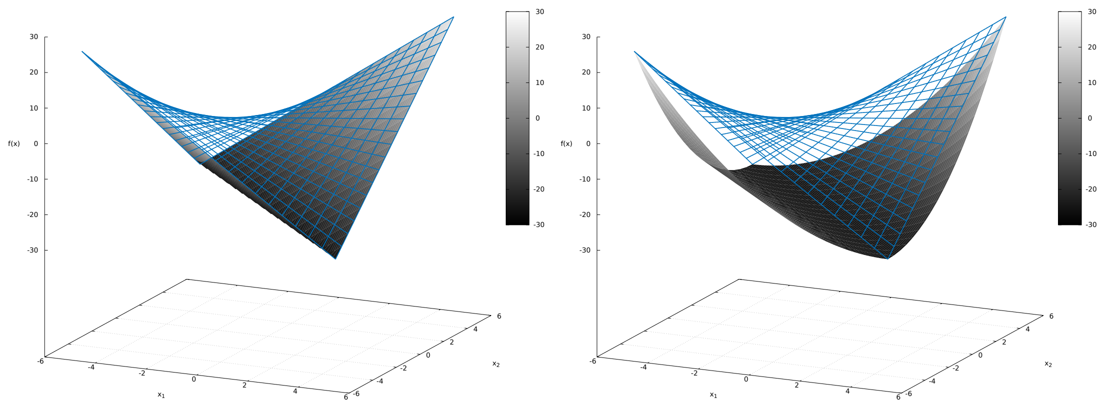


Figure 3.2: Comparison of an exact convex relaxation (left) of the bilinear function $f(x) = x_1 x_2$ with the corresponding αBB relaxation (right).

Specializing the general spatial branch-and-bound in Algorithm 1 with the αBB relaxation above leads to the αBB algorithm stated in Algorithm 2.
To give an illustration of this method to obtain globally optimal solutions, Figure 3.3 shows the αBB algorithm applied to the nonconvex function $f(x) = x^4 - x^2 - 0.05x$ using already the plot capabilities of our software package presented in Chapter 5, while setting the number of state variables and constraints to zero.

3.5 Eigenvalues of an Interval Matrix

As seen in the last section, a crucial element when using αBB is to obtain bounds on the eigenvalues of the interval Hessian. [SW14] compares methods to obtain such bounds and concludes that the estimators based on Gershgorin's circle theorem still yield excellent results in comparison to other proposed methods. The theorem in [Ger31] states the following:

Algorithm 2: An αBB algorithm for global optimization, based on the basic branch-and-bound Algorithm 1 and specialized by the αBB relaxation.

```
Let $\underline{\phi}$ and $\overline{\phi}$ denote the lower and upper bound on the global optimum of the
objective function of an NLP (3.1), $v \in V_0$ the initial compact domain of $v$ and $\epsilon$ a
desired accuracy for the global optimum.
Solve NLP locally on $V_0$ and obtain solution $v_0^*$ and objective value $\phi(v_0^*)$.
Obtain $\alpha_0$ on $X_0$.
Relax NLP on $X_0$ using $\alpha_0$ and obtain convex problem $\mathrm{NLP}_0^{\mathrm{cv}}$.
Solve $\mathrm{NLP}_0^{\mathrm{cv}}$ locally (=globally) on $V_0$ and obtain solution $v_0^{\mathrm{cv},*}$ and objective value
$\phi_0^{\mathrm{cv}}(v_0^{\mathrm{cv},*})$.
Initialize a set of pairs of domains with corresponding local lower bounds
$\mathcal{V} := \{(V_0, \phi_0^{\mathrm{cv}}(v_0^{\mathrm{cv},*}))\}$.
Initialize bounds $\underline{\phi} := {\phi_0}^{\mathrm{cv}}(v_0^{\mathrm{cv},*})$, $\overline{\phi} := \phi(v^*)$, currently best solution $v^* := v_0^*$ and an
iteration counter $i := 0$ that serves as an index only.
while $\mathcal{V} \neq \emptyset$ do
    if $\overline{\phi} - \underline{\phi} \leq \epsilon$ then
        Stop algorithm, global solution obtained.
    end
    Increment iteration counter $i := i + 1$.
    Select pair $(V_i, \phi^{\mathrm{cv}}(v_i^{\mathrm{cv},*})) \in \mathcal{V}$.
    Branch domain $V_i$ into $N$ disjoint nonempty subdomains $W_{i,1}, \dots W_{i,N}$ such that
    $V = \bigcup_{k \in \{1,\dots,N\}} W_{i,k}$.
    for $k = 1$ to $N$ do
        Solve NLP locally on $W_{i,k}$ and obtain solution $w_{i,k}^*$ and objective value $\phi(w_{i,k}^*)$.
        if $\phi(w_{i,k}^*) < \overline{\phi}$ then
            Update upper bound $\overline{\phi} := \phi(w_{i,k}^*)$.
            Update best known solution $v^* := w_{i,k}^*$.
        end
        Obtain $\alpha_{i,k}$ on $W_{i,k}$.
        Relax NLP on $W_{i,k}$ using $\alpha_{i,k}$ and obtain convex problems $\mathrm{NLP}_{i,k}^{\mathrm{cv}}$.
        Solve $\mathrm{NLP}_{i,k}^{\mathrm{cv}}$ locally (=globally) on $W_{i,k}$ and obtain solution $w_{i,k}^{\mathrm{cv},*}$ and objective
        value $\phi_k^{\mathrm{cv}}(w_{i,k}^{\mathrm{cv},*})$.
    end
    Set $\mathcal{V} := \left(\mathcal{V} \setminus \left(V_i, \phi^{\mathrm{cv}}(v_i^{\mathrm{cv},*})\right)\right) \bigcup_{k \in \{1,\dots,N\}} \{(W_{i,k}, \phi_{i,k}^{\mathrm{cv}}(w_{i,k}^{\mathrm{cv},*}))\}$.
    Set new global lower bound $\underline{\phi} := \min\{\phi^{\mathrm{cv}}(v^{\mathrm{cv},*}) : (V, \phi^{\mathrm{cv}}(v^{\mathrm{cv},*})) \in \mathcal{V}\}$ .
    Fathom: remove all pairs with worse local lower bound than the global upper
    bound, i.e. $\mathcal{V} := \mathcal{V} \setminus \{\phi^{\mathrm{cv}}(v^{\mathrm{cv},*}) : (V, \phi^{\mathrm{cv}}(v^{\mathrm{cv},*})) \in \mathcal{V}, \overline{\phi} \leq \phi^{\mathrm{cv}}(v^*)\}$ .
end
The true global optimum is now bounded: $\phi(v_{\mathrm{glo}}^*) \in [\underline{\phi}, \overline{\phi}]$ with $\mathrm{w}([\underline{\phi}, \overline{\phi}]) \leq \epsilon$ .
Furthermore, the best solution found $v^*$ is either the global optimum or has an
objective function value within the desired accuracy.
```

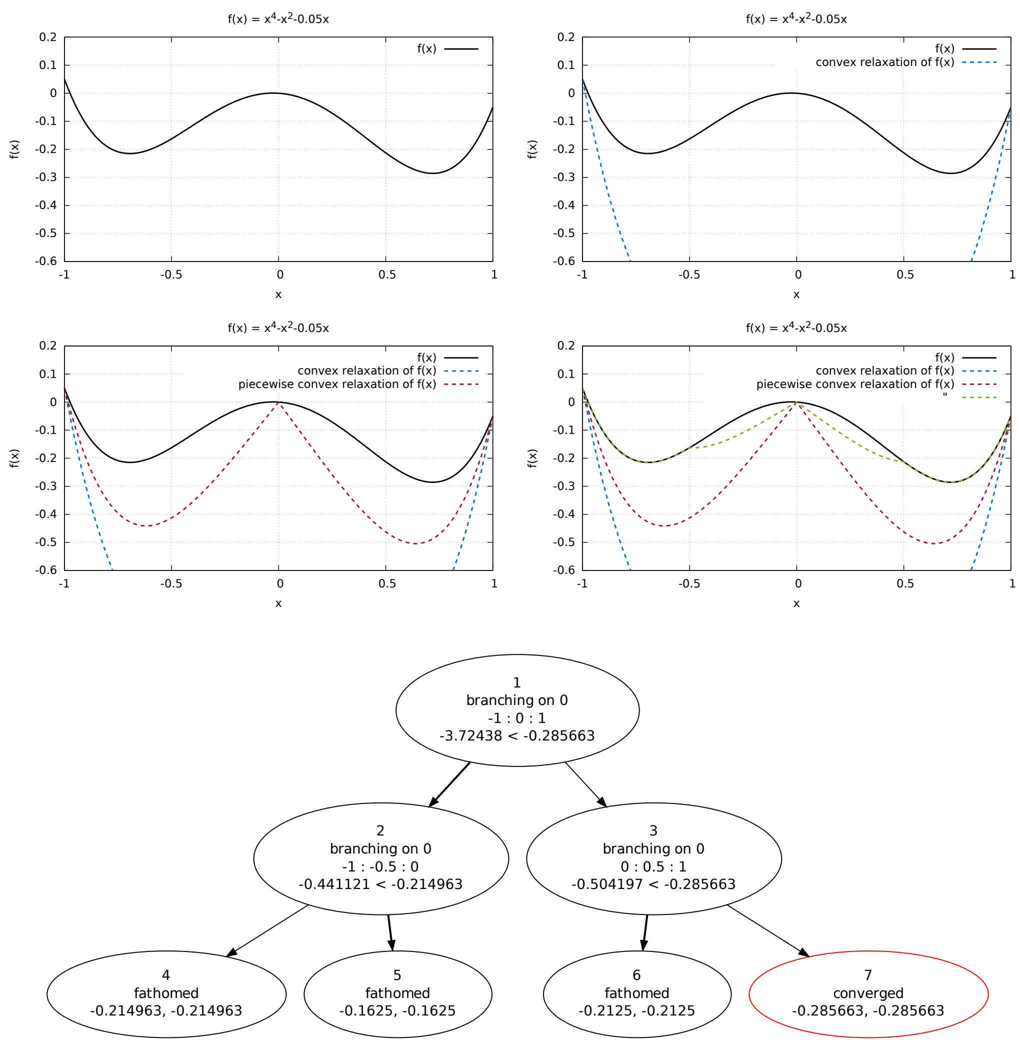


Figure 3.3: Above is an example application of the αBB on a one dimensional non convex function. The four plots illustrate how the branching tightens the convex relaxations till a convergence is achieved. Below is a visualization of the resulting branch-and-bound tree with the topmost number indicating the node index. The second line in each node describes the chosen branching index, which is always 0 in this one dimensional example. The third line gives the local lower and upper bounds of the domain of x and the last line states the global lower and upper bound available while processing the node.

Theorem 2 (Gershgorin's circle theorem). *Let $A \in \mathbb{C}^{n\times n}$ with entries a_{ij}. Let $R_i = \sum_{j\neq i} |a_{ij}|$ be the sum of absolute values of the non-diagonal entries in the i-th row. Every eigenvalue of A lies within at least one of the Gershgorin discs $D(a_{ii}, R_i) := \{z : \|z - a_{ii}\|_2 \subseteq R_i$ centered at a_{ii} with radius R_i.*

By assuming the worst case, it is possible to transfer this theorem to the case of real interval matrices.

Theorem 3 (Gershgorin's circle theorem for interval matrices)**.** *Let $A \in R^{N \times N}$ with interval entries A_{IJ}. Let $R_i = \sum_{j \neq i} |a_{ij}|$ be the sum of absolute values of the non-diagonal entries in the i-th row. Then for all eigenvalues $\lambda_j(A)$ $\exists i$, such that $\lambda_j(A) \in \bigcup_{a \in A_{ii}} D(a, R_i)$.*

In other words, the eigenvalues now lie in the union of infinite many Gershgorin discs with their center being in the corresponding intervals A_{ii}.
Figures 3.4, 3.5 and 3.6 show an illustration of both theorems and a visualization of the idea behind αBB as seen in a talk by Westerlund [Wes12].

$$\begin{pmatrix} 2+i & 2 & -1 \\ 1 & 5 & i \\ 1 & -1 & -1 \end{pmatrix}$$

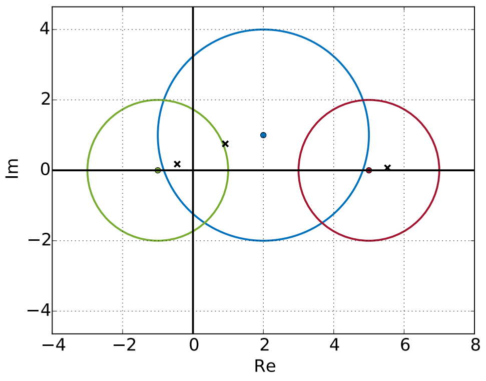


Figure 3.4: Illustration of Gershgorin's circle theorem.

$$\begin{pmatrix} [2,5] & [-1,3] & 0 \\ [-1,3] & [5,6] & [-1,0] \\ 0 & [-1,0] & [-2,-1] \end{pmatrix}$$

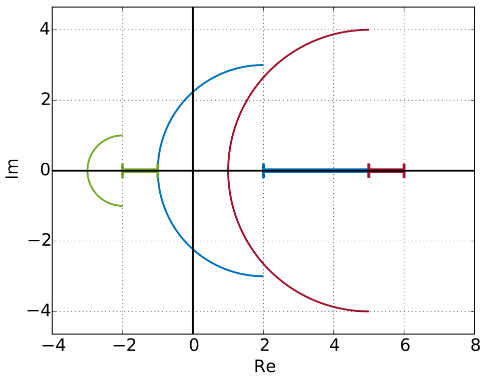


Figure 3.5: Illustration of Gershgorin's circle theorem in the interval version.

$$\begin{pmatrix} [2,5]+1 & [-1,3] & 0 \\ [-1,3] & [5,6]+0 & [-1,0] \\ 0 & [-1,0] & [-2,-1]+3 \end{pmatrix}$$

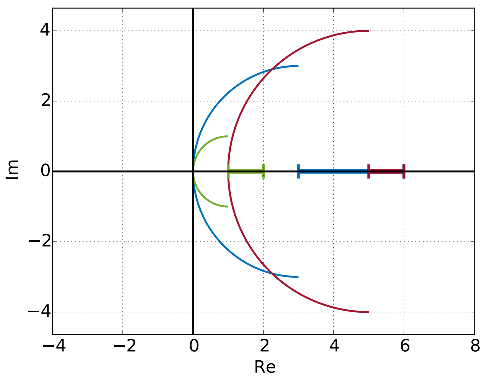


Figure 3.6: Illustration of a diagonal shift using Gershgorin's circle theorem.

Therefore, doing a diagonal shift $\nabla^2 f + \mathrm{diag}(\alpha)$ of the Hessian using

$$\alpha_i = -\frac{1}{2}\min\left(0, \underline{a}_{ii} - \sum_{j\neq i} |a_{ij}|\right) \tag{3.20}$$

guarantees that all eigenvalues of f^{cv} are nonnegative and thus f^{cv} is indeed convex.
There are many other approaches including nondiagonal shifts [SWMF12], but according to [SW14], the scaled diagonal Gerschgorin method works reasonably well in comparison to other methods on a range of test functions. This scaled version that we use in our multiple-shooting-based algorithm in Chapter 4 is

$$\alpha_i = -\frac{1}{2}\min\left(0, \underline{a}_{ii} - \sum_{j\neq i} |a_{ij}|\frac{d_j}{d_i}\right) \tag{3.21}$$

with a scaling vector $d := \overline{x} - \underline{x}$.
Concave overestimations are important for the relaxation for equality constraints such as the direct multiple shooting matching conditions shown in Equation (2.6). We can overestimate the largest eigenvalue accordingly with

$$\alpha_i = \frac{1}{2}\max\left(0, \overline{a}_{ii} + \sum_{j\neq i} |a_{ij}|\right) \tag{3.22}$$

and

$$\alpha_i = \frac{1}{2}\max\left(0, \overline{a}_{ii} + \sum_{j\neq i} |a_{ij}|\frac{d_j}{d_i}\right). \tag{3.23}$$

3.6 Validated Integration

The discretized OCP from Equation (2.8) contains state variables. Therefore, one integral part in determining the global optimum of an OCP is to determine bounds of the states. In fact, a repeated forward propagation of the state bounds and a global optimization of the objective function using those bounds is already a method to globally optimize OCPs, as shown for example in [LS07a] and briefly described in Algorithm 3. In contrast, more complex global optimal control algorithms, such as the one presented in the following chapters, try to tighten those bounds through additional calculations, often leading to a faster convergence of the BB method.
In contrast to traditional integrators with a variable step size that usually only approximate the local [Alb10] or global [Bei12] error to determine a suitable time step [Sha05], validated integrators enclose the solution trajectories using interval-arithmetic-based Taylor series expansions and the corresponding remainder term. This is often done in two steps. First, a cheap a priori enclosure is generated together with a step size such that the solution exists and is unique. In a second phase, this a priori bound is tightened. According to the survey [Rih94], the idea goes back to [Moo66]. [Ned06] is a more recent survey focusing on the differences in the software implementations that were developed. A different approach in [Sco12] uses McCormick relaxations to improve the bounds of a cheap a priori bound esti-

mation and [HVC13] finally reverses the following two-phase approach by constructing the predictor first and then determining the step size. The authors in [VHC15] present a unified framework for such enclosures of the state variables.

Although we do not contribute to this extensive field, we employ the software packages `VNODE` [NJP01] and `VSPODE` [LS07b]. Due to the differences, the first one is a potential method to calculate the bounds even for larger problems, whereas the latter one usually produces tighter bounds, but scales worse for an increase in the control discretization and control value dimension, because the control is represented by a Taylor model as well. See Chapter 6 for a detailed comparison in some of the numerical examples. Therefore, this section gives a brief overview over the idea behind the two approaches and highlights the differences.

First, we need the Taylor coefficients of a sufficiently often continuously differentiable function f. Written in recursive form we obtain

$$f^{[0]}(x) = x \tag{3.24a}$$

$$f^{[i]}(x) = \frac{1}{i}\left(\frac{\partial f^{[i-1]}}{\partial x} f\right)(x) \quad \forall i \geq 1 \quad . \tag{3.24b}$$

These coefficients can be determined by means of automatic differentiation (AD) [GW08] using software implementations such as `CppAD` [Bel] or `FADBAD++` [BS96].

A common method to generate coarse a priori bounds for the next time interval $[t_j, t_{j+1}]$ and obtaining a step size $h_j = t_{j+1} - t_j$ is to use first ($i = 1$) or even high-order enclosures [NJP01]. Given a desired maximum on the domain $\tilde{X}_j$ of the enclosure, we choose $0 < h_j \leq t_N$ such that

$$x_j + \sum_{i=1}^{k-1} h_j^i f^{[i]}(x_j) + h_j^k f^{[k]}(\tilde{X}_j) \subseteq \tilde{X}_j,\ \forall t \in [t_j, t_{j+1}] \tag{3.25}$$

with $k \geq 1$ and $\forall x_j \in \tilde{X}_j$, then there exists a unique solution on $[t_j, t_{j+1}]$ and

$$x(t) \in x_j + \sum_{i=1}^{k-1} h_j^i f^{[i]}(x_j) + h_j^k f^{[k]}(\tilde{X}_j)\,. \tag{3.26}$$

In a second phase, these state bounds are tightened using either an interval approximation of the remainder using the mean-value evaluation as implemented in `VNODE`

$$x_j + \sum_{i=1}^{k-1} (t - t_j)^i f^{[i]}(x_j) + h_j^k f^{[k]}(X_j) \tag{3.27a}$$

$$\begin{aligned} &\subseteq x_j + \sum_{i=1}^{k-1} (t - t_j)^i f^{[i]}(x_j) + h_j^k f^{[k]}(X_j) \\ &\quad + \left(I + \sum_{i=1}^{k-1} h_j^i \frac{\partial f^{[i]}}{\partial x}(X_j)\right)(X_j - x_j) \quad \forall t \in [t_j, t_{j+1}],\ \forall x_j, \tilde{x}_j \in X_j \end{aligned} \tag{3.27b}$$

or a Taylor model as in `VSPODE`, where a full interval polynomial approximation representing the enclosure overestimation is propagated using operator overloading and evaluated at the desired time steps.

Both variants allow even for contracting enclosures which means that the width of the state bounds is not increasing and may even shrink over time. Nevertheless, special care has to be taken about the so-called wrapping effect that occurs because the remainder term may not be accurately represented by box intervals aligned to the coordinate axes. This effect is one of the sources for exponentially increasing interval widths that may be observed. One of the first and often applied remedies is Lohner's QR factorization [Loh92] that transforms coordinate-axis-aligned boxes into a better fitting parallelepiped, effectively reducing the overestimation in each step.

Using validated integration to generate bounds on the states, we can derive a first single shooting algorithm to solve OCPs as seen in (2.1) with an underlying IVP problem. We alternate between propagating the states with respect to bounds on the control parametrization q_i and the control values p using the resulting interval at the final time t_N, we can evaluate the objective function and use this as a lower bound on the global optimum. Algorithm 3 formalizes this method.

Finally, we can now use the validated integration to derive bounds on the second-order sensitivities and thus relax the solution of an ODE using αBB. Combining Algorithms 2 with an validated integrator, we can derive the single-shooting-based αBB algorithm for OCPs shown in Algorithm 4.

Algorithm 3: A branch-and-bound-based algorithm for IVP-based OCPs using only a validated integrator.

```
Let $\underline{\Phi}$ and $\overline{\Phi}$ denote the lower and upper bound on the global optimum of the objective
function of an IVP-based discretized OCP (2.4), $(q,p) \in P_0$ the initial domain of the
control discretization $q$ and control values $p$, and $\epsilon$ a desired accuracy for the global
optimum.
**Simulate** OCP on $P_0$ using a validated integrator and obtain bounds $X_0$ on the final
state, such that $x(t_N;(q_0,p_0)) \in X_0 \quad \forall (p_0,q_0) \in X_0$.
**Evaluate** the objective function on $P_0$ using natural interval extension and obtain
bounds $\Phi_0 := \Phi_{\mathrm{M}}(X_0,P_0)$.
**Initialize** a set of pairs of domains with corresponding local lower bounds on the
objective function $\mathcal{P} := \{(P_0, \underline{\Phi_0}\}$.
**Initialize** bounds $\underline{\Phi} := \underline{\Phi_0}$, $\overline{\Phi} := \overline{\Phi_0}$ and an iteration counter $i := 0$ that serves as an
index only.
**while** $\mathcal{P} \neq \emptyset$ **do**
    **if** $\overline{\Phi} - \underline{\Phi} \leq \epsilon$ **then**
        **Stop** algorithm, global solution obtained.
    **end**
    **Increment** iteration counter $i := i+1$.
    **Select** pair $(P, \underline{\Phi}) \in \mathcal{P}$.
    **Branch** domain $P$ into $N$ disjoint nonempty subdomains $Q_{i,1}, \dots Q_{i,N}$ such that
    $P = \bigcup_{k \in \{1,\dots,N\}} Q_{i,k}$.
    **for** $k = 1$ **to** $N$ **do**
        **Simulate** OCP on $Q_{i,k}$ using a validated integrator and obtain bounds $X_{i,k}$,
        such that $x(t_N;(q_{i,k},p_{i,k})) \in X_{i,k} \quad \forall (q_{i,k},p_{i,k}) \in Q_{i,k}$.
        **Evaluate** the objective function on $Q_{i,k}$ using natural interval extension,
        obtaining bounds $\Phi_{i,k} := \Phi_{\mathrm{M}}(X_{i,k},P_{i,k})$.
        **if** $\overline{\Phi}_{i,k} < \overline{f}$ **then**
            **Update** upper bound $\overline{f} := \overline{\Phi}_{i,k}$.
        **end**
    **end**
    **Set** $\mathcal{P} := \left(\mathcal{P} \setminus \left(P, \underline{\Phi}\right)\right) \bigcup_{k \in \{1,\dots,N\}} \{(Q_{i,k}, \underline{\Phi}_{i,k})\}$.
    **Set** new global lower bound $\underline{\Phi} := \min\{\underline{\Phi} : (P, \underline{\Phi}) \in \mathcal{P}\}$ .
    **Fathom:** remove all pairs with worse local lower bound than the global upper
    bound, i.e. $\mathcal{P} := \mathcal{P} \setminus \{\underline{\phi} : (P, \underline{\phi}) \in \mathcal{P}, \overline{\Phi} \leq \underline{\phi}\}$ .
**end**
The true global optimum is now bounded: $\Phi^*_{\mathrm{glo}} \in [\underline{\Phi}, \overline{\Phi}]$ with $\mathrm{w}([\underline{\Phi}, \overline{\Phi}]) \leq \epsilon$ .
Furthermore, any solution remaining in $\mathcal{P}$ has an objective function value within the
desired accuracy.
```

Algorithm 4: A single-shooting-based αBB algorithm for global optimal control that builds upon on the basic branch-and-bound Algorithm 1 and is specialized by the αBB relaxation using an validated integrator to obtain bounds on the second-order sensitivities.

```
Let $\underline{\Phi}$ and $\overline{\Phi}$ denote the bounds on the globally optimal objective value of an
discretized OCP (2.4), $(q, p) \in P_0$ the initial domain of the control discretization $q$ and
control values $p$, and $\epsilon$ a desired accuracy for the global optimum.

**Solve** OCP locally on $P_0$ and obtain solution $(q_0^*, p_0^*)$ and $\Phi_\mathrm{M}(x(t_N), p_0)$.
**Simulate** OCP on $P_0$ using a validated integrator along with the first (Equation (2.9))
and second (Equation (2.14)) order variational differential equations to obtain bounds
$X_0$ on the final state and $H_0$ on the second-order sensitivities.
**Relax** OCP on $P_0$ using an $H_0$ based $\alpha_0$ and obtain convex problem $\mathrm{OCP}_0^{\mathrm{cv}}$.
**Solve** $\mathrm{OCP}_0^{\mathrm{cv}}$ locally (=globally) on $P_0$ and obtain solution $(q_0^{\mathrm{cv},*}, p_0^{\mathrm{cv},*})$ and objective
value $\Phi_{\mathrm{M},0}(x(t_N), p_0)$.
**Initialize** a set of pairs of domains with corresponding local lower bounds
$\mathcal{P} := \{(P_0, \Phi_{\mathrm{M},0}(x(t_N), p_0))\}$.
**Initialize** bounds $\underline{\Phi} := \Phi_{\mathrm{M},0}(x(t_N), p_0)$, $\overline{\Phi} := \Phi_\mathrm{M}(x(t_N), p_0)$, currently best solution
$(q^*, p^*) := (q_0^*, p_0^*)$ and an iteration counter $i := 0$.
**while** $\mathcal{P} \neq \emptyset$ **do**
    **if** $\overline{\Phi} - \underline{\Phi} \le \epsilon$ **then**
        **Stop** algorithm, global solution obtained.
    **end**
    **Increment** iteration counter $i := i + 1$.
    **Select** pair $(P, \Phi_\mathrm{M}(x(t_N), p)) \in \mathcal{P}$.
    **Branch** domain $P$ into $N$ disjoint nonempty subdomains $Q_1, \dots Q_N$ such that
    $Q = \bigcup_{k \in \{1,\dots,N\}} Q_k$.
    **for** $k = 1$ **to** $N$ **do**
        **Solve** OCP locally on $Q_k$ obtaining $(q_{i,k}^{\mathrm{cv},*}, p_{i,k}^{\mathrm{cv},*})$ and $\Phi_{\mathrm{M},i,k}(x(t_N), p_{i,k})$.
        **if** $\Phi_{M,i,k}(x(t_N), p_{i,k}) < \overline{\Phi}$ **then**
            **Update** upper bound $\overline{\Phi} := \Phi_{\mathrm{M},i,k}(x(t_N), p_{i,k})$ and best known solution
            $(q^*, p^*) := (q_{i,k}^*, p_{i,k}^*)$.
        **end**
        **Simulate** OCP on $Q_{i,k}$ with validated integration to obtain $X_{i,k}$ and $H_{i,k}$.
        **Relax** OCP on $Q_{i,k}$ using an $H_{i,k}$ based $\alpha_{i,k}$ and obtain convex $\mathrm{OCP}_{i,k}^{\mathrm{cv}}$.
        **Solve** $\mathrm{OCP}_{i,k}^{\mathrm{cv}}$ locally (=globally) on $Q_{i,k}$ and obtain solution $(q_{i,k}^{\mathrm{cv},*}, p_{i,k}^{\mathrm{cv},*})$ and
        objective value $\Phi_{\mathrm{M},i,k}(x(t_N), p_{i,k})$.
    **end**
    **Set** $\mathcal{P} := (\mathcal{P} \setminus (P, \Phi_\mathrm{M}(x(t_N), p))) \bigcup_{k \in \{1,\dots,N\}} (Q_k, \Phi_{\mathrm{M},i,k}(x(t_N), p_{i,k}))$.
    **Set** new global lower bound $\underline{\Phi} := \min\{\Phi_\mathrm{M}(x(t_N), p) : (P, \Phi_\mathrm{M}(x(t_N), p)) \in \mathcal{P}\}$ .
    **Fathom:** remove all pairs with worse local lower bound than the global upper
    bound, i.e. $\mathcal{P} := \mathcal{P} \setminus \{\Phi_\mathrm{M}(x(t_N), p) : (P, \Phi_\mathrm{M}(x(t_N), p)) \in \mathcal{P}, \overline{\Phi} \le \Phi_\mathrm{M}(x(t_N), p)\}$ .
**end**
The true global optimum is now bounded: $\Phi_\mathrm{M}(x(t_N), p_{\mathrm{glo}}^*) \in [\underline{\Phi}, \overline{\Phi}]$ with
$\mathrm{w}([\underline{\Phi}, \overline{\Phi}]) \le \epsilon$. Furthermore, the best solution found $(q^*, p^*)$ is either the global
optimum or has an objective function value within the desired accuracy.
```

Part II

Mathematical Method

Chapter 4

Global Optimal Control Using Multiple Shooting

We present a novel and efficient method to obtain deterministic global solutions for optimal control problems (OCPs) by applying the α-Branch-and-Bound (αBB) method [AMF95, AF96] in a direct multiple shooting framework [BP84]. On the one hand, we obtain a larger problem with additional variables and equality constraints, on the other hand, we will prove that the resulting convex relaxations of the reformulated problem are at least as good as the corresponding relaxations of the original formulation and a true improvement in nontrivial cases. This property is due to the observation that the variational differential equations have a constant initial value on each multiple shooting node leading to a certain decoupling by this reformulation. We prove that using αBB leads to equal or smaller α values for the convex underestimation of the differential equations and hence may improve the lower bounds on each node.
In extension to previous literature [EF00a, EF00b, PA04, PA05] that describes direct single-shooting-based approaches and how to treat pointwise equality constraints involving the state variables during the integration horizon, we propose for the first time to use a multiple shooting formulation instead to improve the convex relaxations. Numerical results indicate that the computational improvement obtained due to the tighter convexifications are not only able to compensate for the additional variables, but to improve the overall computational effort. Furthermore, the direct multiple shooting approach still maintains all other advantages compared to direct single shooting, such as improved stability and efficient parallelization.
To account for the additional variables, we will introduce a specialized treatment of the additional variables in the multiple shooting case and give a proof that these variables do not increase the size of the branch-and-bound tree using this strategy and thus the computational overhead reduces to a slightly larger, but highly structured NLP on each tree node.
In the last part of this chapter, we combine our theoretical results to formulate the full multiple-shooting-based αBB algorithm that is implemented in our software package that will be introduced in Chapter 5 and is used to obtain the numerical results in Chapter 6 that will indicate that this novel lifting is indeed a viable strategy that leads to a better performance in comparison with the usually applied direct single shooting method.

4.1 Treatment of Matching Conditions

Recalling the discretized forms of an OCP from Chapter 3 in Equation (2.4) for direct single shooting and Equation (2.8) for direct multiple shooting, the major difference lies in the additional auxiliary variables s_i and the corresponding matching conditions as shown in Equation (2.6).

Equality constraints are convex in the linear case only, therefore we need to split up the matching conditions into two inequality constraints with opposite signs:

$$x(t_{i+1}; s_i) - s_{i+1} = 0 \Leftrightarrow \left\{ \begin{matrix} x(t_{i+1}; t_i, s_i, q_i, p) - s_{i+1} \leq 0 \\ -x(t_{i+1}; t_i, s_i, q_i, p) + s_{i+1} \leq 0 \end{matrix} \right\} . \tag{4.1}$$

Adding or subtracting s_{i+1} preserves convexity. Therefore, we just need to convexify

$$x(t_{i+1}; t_i, s_i, q_i, p) \quad \text{and} \quad -x(t_{i+1}; t_i, s_i, q_i, p) , \tag{4.2}$$

the latter being the concave relaxation of $x(t_{i+1}; t_i, s_i, q_i, p)$ which can be obtained through the same methods introduced in Chapter 4 by overestimating the eigenvalues of the interval Hessian and applying a corresponding shift by means of αBB such that all eigenvalues become negative instead of positive for the convexification. To distinguish between the α values, we use the notation $\breve{\alpha}$ for the values necessary to convexify $x(t_{i+1}; t_i, s_i, q_i, p)$ and $\hat{\alpha}$ for the concave relaxation of $x(t_{i+1}; t_i, s_i, q_i, p)$ that is equivalent to the convex relaxation of $-x(t_{i+1}; t_i, s_i, q_i, p)$. Please note that the additional computational overhead in this case is rather low, because using Gershgorins circle theorem, the computationally expensive part is to determine the interval Hessian in the first place and not the over- and underestimation of the eigenvalues.

Applying the αBB relaxation as described in Chapter 3 leads to the relaxed matching conditions

$$\begin{aligned} x_j(t_{i+1}; t_i, s_i, q_i, p) - s_{i+1,j} + \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk} (\bar{s}_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) \\ + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk} (\bar{q}_{i,k} - q_{i,k})(\underline{q}_{i,k} - q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk} (\bar{p}_k - p_k)(\underline{p}_k - p_k) \leq 0 \end{aligned} \tag{4.3}$$

$$\begin{aligned} -x_j(t_{i+1}; t_i, s_i, q_i, p) + s_{i+1,j} + \sum_{k=1}^{n_x} \hat{\alpha}^{\mathrm{s}}_{ijk} (\bar{s}_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) \\ + \sum_{k=1}^{n_q} \hat{\alpha}^{\mathrm{q}}_{ijk} (\bar{q}_{i,k} - q_k)(\underline{q}_{i,k} - q_{i,k}) + \sum_{k=1}^{n_p} \hat{\alpha}^{\mathrm{p}}_{ijk} (\bar{p}_k - p_k)(\underline{p}_k - p_k) \leq 0 \end{aligned} \tag{4.4}$$

for all multiple shooting nodes indexed by $i = 0, \ldots, N-1$ and all components of x and s indexed by $j = 0, \ldots, n_x$ and with $[\underline{s}, \bar{s}]$, $[\underline{q}, \bar{q}]$, $[\underline{p}, \bar{p}]$ being the bounds on s, q, p.

Based on Equation (2.8), we obtain the full discretized OCP:

$$\min_{s,q,p} \quad \Phi_{\mathrm{M}}(s_N,p)+\sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \Phi_{\mathrm{L}}(t,x(t;t_i,s_i,q_i,p),\tilde{u}_i(t,q_i),p)\,\mathrm{d}t \tag{4.5a}$$

$$\text{s.t.} \quad 0 \geq x_j(t_{i+1};t_i,s_i,q_i,p)-s_{i+1,j}+\sum_{k=1}^{n_x}\breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \tag{4.5b}$$

$$+\sum_{k=1}^{n_q}\breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k})+\sum_{k=1}^{n_p}\breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k) \tag{4.5c}$$

$$0 \geq -x_j(t_{i+1};t_i,s_i,q_i,p)+s_{i+1,j}+\sum_{k=1}^{n_x}\hat{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \tag{4.5d}$$

$$+\sum_{k=1}^{n_q}\hat{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_k)(\underline{q}_{i,k}-q_{i,k})+\sum_{k=1}^{n_p}\hat{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k) \tag{4.5e}$$

$$0 \leq c^{\mathrm{cc}}(t_i,s_i,\tilde{u}_i(t_i,q_i),p) \tag{4.5f}$$

$$0 \leq r^{\mathrm{eq,cc}}(s_0,\dots,s_N,p) \tag{4.5g}$$

$$0 \leq -r^{\mathrm{eq,cv}}(s_0,\dots,s_N,p) \tag{4.5h}$$

$$0 \leq r^{\mathrm{ieq,cc}}(s_0,\dots,s_N,p) \tag{4.5i}$$

$$\forall i \in \{0,\dots,N-1\}, j \in \{1,\dots,n_x\} \tag{4.5j}$$

with $c^{\mathrm{cc}}(t_i,s_i,\tilde{u}_i(t_i,q_i),p)$ being a concave relaxation of c, $r^{\mathrm{eq,cv}}$ and $r^{\mathrm{eq,cc}}$ being convex respective concave relaxations of r^{eq} and $r^{\mathrm{ieq,cc}}$ being a concave relaxation of r^{ieq}. These may be generated using αBB as well if necessary.

Since we do not always have to distinguish between α^s, α^q and α^p, we define

$$\breve{\alpha}=\begin{pmatrix}\breve{\alpha}^s\\ \breve{\alpha}^q\\ \breve{\alpha}^p\end{pmatrix}, \quad \hat{\alpha}=\begin{pmatrix}\hat{\alpha}^s\\ \hat{\alpha}^q\\ \hat{\alpha}^p\end{pmatrix} \quad \text{and} \quad \alpha=\begin{pmatrix}\breve{\alpha}\\ \hat{\alpha}\end{pmatrix} \tag{4.6}$$

to ease the notation.

4.2 Theoretical Comparison

In this section, we begin with a theoretical comparison of the quality of the αBB convexifications between direct single and multiple shooting. We have to take care of potentially different integration steps and therefore, most results in the first part have one exact version, assuming exact integration and a numerical version taking into account the numerical integration with an additional assumption concerning the integration steps. In the second part, we focus on the treatment of the additional variables introduced by our method and the potential impact on the computational time, introducing a specialized branching scheme that eliminates the necessity to branch on those extra variables.

4.2.1 Comparison with Direct Single Shooting

To compare the convexifications of direct single and direct multiple shooting and to ease the formulation of the following lemmata, theorems and corollaries, we introduce the assumption that, apart from this fundamental difference, all other means of the αBB relaxation, in particular the method to overestimate α is identical in both cases.

Assumption 4. *We assume that we have an OCP as seen in Equation* (2.1) *with an at least twice continuously differentiable right-hand side (RHS) function $f(t,x(t),u(t),p)$ together with a control discretization $\tilde{u}_i(t,q_i)$ and αBB-based relaxations of all nonconvex functions. Furthermore, we overestimate α based on Gershgorin's circle theorem as shown in Equation* (3.20).

We begin this comparison for linear objective functions of Mayer-type. Many test problems and applications in the global optimal control literature are formulated in this way, for example compare Problems (6.3) and (6.7) in Chapter 6. Furthermore, we can always obtain such a type by combining any nonlinear Mayer type and Lagrange type objective function into a new Lagrange objective $\Phi_{\mathrm{L}'} = \Phi_{\mathrm{L}} + \nabla\Phi_{\mathrm{M}}$ as shown in [Ces83] among others. Then we can transform this combined Lagrange type into Mayer type introducing a new state $x_{n_x+1} = \Phi_{L'}$. The resulting Mayer type objective function is now $\Phi_{M'} = x_{n_x+1}(t_N)$ and thus linear in the new state, effectively hiding any nonlinearities of the objective function in the additional state variable. Please note that this reformulation and relaxing the problem do not commute due to our relaxations being based on interval arithmetic. Therefore, this reformulation may not lead to an equivalent convex relaxation.
Without loss of generality, let us assume that the initial value is fixed. In the single shooting case, such free initial values would be included in an augmented parameter vector p' and we are able to identify any free initial value in s_0 with these additional components.
We abbreviate the following Lemma, we combine both assumptions formally into

Assumption 5. *We assume that the given OCP has a linear objective function of Mayer-type and fixed initial values $x(t_0)$.*

To compare the quality of the αBB-based convex relaxations of the direct single and the multiple shooting discretization with each other, we first compare the single shooting relaxation with the two point multiple shooting case, where only one node at the final time t_1 is introduced.
We formalize this result that was observed numerically in [PA02] and proof that under Assumptions 4 and 5, introducing a multiple shooting node at the final time t_1 leads to convex relaxation that is at least as good as the one obtained using Algorithm 4.

Lemma 6 (Comparison of DSS with Two Point DMS with a Linear Objective Function)**.** *Under Assumptions 4 and 5, using direct single shooting and using direct multiple shooting with only two multiple shooting nodes and corresponding relaxed objective function values Φ_{M}^{dss} and Φ_{M}^{dms}, it holds true that $\Phi_{\mathrm{M}}^{dms}(s_1^*, p^*) \geq \Phi_{\mathrm{M}}^{dss}(x(t_N), p^\star)$ for the optimal solution (s_1^*, q^*, p^*), respectively $(q^\star, p^\star)$.*

Proof. To ease the notation, we define $\tilde{p} := (q^T, p^T)^T$ and obtain for the second derivative of the single shooting objective function with respect to the discretized controls and control values:

$$\frac{\mathrm{d}^2}{\mathrm{d}\tilde{p}^2}\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p) = \frac{\mathrm{d}}{\mathrm{d}\tilde{p}}\left(\frac{\partial\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p)}{\partial x}\frac{\mathrm{d}x(t_N)}{\mathrm{d}\tilde{p}} + \frac{\partial\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p)}{\partial \tilde{p}}\right) \tag{4.7a}$$

$$\begin{aligned} &= \left(\left(\frac{\mathrm{d}x(t_N)}{\mathrm{d}\tilde{p}}\right)^T \frac{\partial^2\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p)}{\partial x^2} + 2\frac{\partial^2\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p)}{\partial x\partial\tilde{p}}\right)\frac{\mathrm{d}x(t_N)}{\mathrm{d}\tilde{p}} \\ &\quad + \left(\frac{\partial\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p)}{\partial x}\otimes I^{n_{\tilde{p}}}\right)\frac{\mathrm{d}^2x(t_N)}{\mathrm{d}\tilde{p}^2} + \frac{\partial^2\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p)}{\partial\tilde{p}^2} \quad \forall\tilde{p}\;. \end{aligned} \tag{4.7b}$$

Due to the objective function being linear by assumption, the second-order partial derivatives of the objective function vanish. Furthermore, the objective function can be written as $c^T x(t_N) + d^T p$ with constant vectors $c \in \mathbb{R}^{n_x}$ and $d \in \mathbb{R}^{n_p}$ and we obtain

$$\frac{\mathrm{d}^2}{\mathrm{d}\tilde{p}^2}\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p) = \left(\frac{\partial\Phi_{\mathrm{M}}^{\mathrm{dss}}(x(t_N),p)}{\partial x}\otimes I^{n_{\tilde{p}}}\right)\frac{\mathrm{d}^2x(t_N)}{\mathrm{d}\tilde{p}^2} \tag{4.8a}$$

$$= \sum_{i=1}^{n_x}(c_i I^{n_{\tilde{p}}})\frac{\mathrm{d}^2x_i(t_N)}{\mathrm{d}\tilde{p}^2} \quad \forall\tilde{p}\;. \tag{4.8b}$$

To ease the following notation, we use a tensor index notation for the individual interval valued elements of the second-order sensitivity bounds and define

$$H_{ijk} := \left(\frac{\mathrm{d}^2x_i(t_N)}{\mathrm{d}\tilde{p}^2}\right)_{j,k} \quad \forall i\in\{1,\ldots,n_x\} \quad \forall j,k\in\{1,\ldots n_{\tilde{p}}\}\;. \tag{4.9}$$

Using this definition and Gershgorin's circle theorem to generate lower bounds on the eigenvalues of the second derivative and calculating α^{dss} as seen in (3.20), we obtain

$$\alpha_j^{\mathrm{dss}} = -\frac{1}{2}\min\left(0, \underline{(\sum_{i=1}^{n_x}c_iH_{ijj})} - \sum_{\substack{k=1\\k\neq j}}^{n_{\tilde{p}}}|\sum_{i=1}^{n_x}c_iH_{ijk}|\right) \tag{4.10a}$$

$$\begin{aligned} &= -\frac{1}{2}\sum_{\substack{i=1\\c_i\geq 0}}^{n_x}c_i\min\left(0, \underline{H}_{ijj} - \sum_{\substack{k=1\\k\neq j}}^{n_{\tilde{p}}}|H_{ijk}|\right) \\ &\quad -\frac{1}{2}\sum_{\substack{i=1\\c_i<0}}^{n_x}c_i\max\left(0, \overline{H}_{ijj} + \sum_{\substack{k=1\\k\neq j}}^{n_{\tilde{p}}}|H_{ijk}|\right)\;. \end{aligned} \tag{4.10b}$$

The objective function is linear by assumption and therefore, the optimal solution lies on the boundary of the feasible set. Using the relaxed matching condition (4.3) with $\check{\alpha}^{\mathrm{dms}}$ and $\hat{\alpha}^{\mathrm{dms}}$ being the corresponding relaxation parameters, we obtain for the single-shooting-based optimal solution $\tilde{p}^\star$:

$$
\begin{aligned}
\Phi_{\mathrm{M}}^{\mathrm{dss}} &= c^T x(t_N) + d^T \tilde{p}^\star + \sum_{j=1}^{n_{\tilde{p}}} \alpha_j^{\mathrm{dss}} (\overline{\tilde{p}} - \tilde{p}^\star)(\underline{\tilde{p}} - \tilde{p}^\star) \qquad \text{(using Equation (4.10))} && \text{(4.11a)}\\
&\quad - \sum_{\substack{i=1\\ c_i<0}}^{n_x} c_i \left(\sum_{j=1}^{n_{\tilde{p}}} \frac{1}{2} \max\left(0, \overline{H}_{i,j,j} + \sum_{\substack{k=1\\ k\neq j}}^{n_{\tilde{p}}} |H_{ijk}| \right) (\overline{\tilde{p}}_j - \tilde{p}_j^\star)(\underline{\tilde{p}}_j - \tilde{p}_j^\star) \right) && \text{(4.11b)}\\
&= c^T x(t_N) + d^T \tilde{p}^\star + \sum_{\substack{i=1\\ c_i\geq 0}}^{n_x} c_i \left(\sum_{j=1}^{n_{\tilde{p}}} \frac{1}{2} \min\left(0, \underline{H}_{i,j,j} - \sum_{\substack{k=1\\ k\neq j}}^{n_{\tilde{p}}} |H_{ijk}| \right) (\overline{\tilde{p}}_j - \tilde{p}_j^\star)(\underline{\tilde{p}}_j - \tilde{p}_j^\star) \right)\\
&\quad - \sum_{\substack{i=1\\ c_i<0}}^{n_x} c_i \left(\sum_{j=1}^{n_{\tilde{p}}} \hat{\alpha}_j^{\mathrm{dms}} (\overline{\tilde{p}}_j - \tilde{p}_j^\star)(\underline{\tilde{p}}_j - \tilde{p}_j^\star) \right) && \text{(4.11c)}\\
&= c^T x(t_N) + d^T \tilde{p}^\star + \sum_{\substack{i=1\\ c_i\geq 0}}^{n_x} c_i \left(\sum_{j=1}^{n_{\tilde{p}}} \breve{\alpha}_j^{\mathrm{dms}} (\overline{\tilde{p}}_j - \tilde{p}_j^\star)(\underline{\tilde{p}}_j - \tilde{p}_j^\star) \right)\\
&\quad + \sum_{\substack{i=1\\ c_i<0}}^{n_x} c_i (x(t_N) - \sum_{j=1}^{n_{\tilde{p}}} \hat{\alpha}_j^{\mathrm{dms}} (\overline{\tilde{p}}_j - \tilde{p}_j^\star)(\underline{\tilde{p}}_j - \tilde{p}_j^\star)) + d^T \tilde{p}^\star && \text{(4.11d)}\\
&= \sum_{\substack{i=1\\ c_i\geq 0}}^{n_x} c_i (x_i(t_N) + \sum_{j=1}^{n_{\tilde{p}}} \breve{\alpha}_j^{\mathrm{dms}} (\overline{\tilde{p}}_j - \tilde{p}_j^\star)(\underline{\tilde{p}}_j - \tilde{p}_j^\star))\\
&= \sum_{i=1}^{n_x} c_i \cdot [x_i(t_N) + \sum_{j=1}^{n_{\tilde{p}}} \breve{\alpha}_j^{\mathrm{dms}} (\overline{\tilde{p}}_j - \tilde{p}_j^\star)(\underline{\tilde{p}}_j - \tilde{p}_j^\star), x(t_N) - \sum_{j=1}^{n_{\tilde{p}}} \hat{\alpha}_j^{\mathrm{dms}} (\overline{\tilde{p}}_j - \tilde{p}_j^\star)(\underline{\tilde{p}}_j - \tilde{p}_j^\star)] + d^T \tilde{p}^\star\\
& && \text{(4.11e)}\\
&= \Phi_{\mathrm{M}}^{\mathrm{dms}}(s_1, \tilde{p}^\star)\,. && \text{(4.11f)}
\end{aligned}
$$

Therefore, $\tilde{p}^\star$ is the optimal solution for $\Phi_{\mathrm{M}}^{\mathrm{dms}}$ if it is feasible. In particular $s_1 \subseteq X(t_N)$ with the state bounds $X(t_N)$ at the final time t_N. Therefore, we can use the constant interval $X(t_N)$ as bounds S_1 on s_1. Adding these constant bounds on s_1 to the resulting NLP can only further tighten the feasible set in the multiple shooting case and thus improve the relaxation $\Rightarrow \Phi_{\mathrm{M}}^{\mathrm{dms}}(s_1^*, p^*) \geq \Phi_{\mathrm{M}}^{\mathrm{dss}}(x(T), \tilde{p}^\star)$. □

In Lemma 6, we assumed a linear objective function. For the comparison of single-shooting-based relaxations of general nonlinear twice continuously differentiable objective functions, the problem becomes the question how the αBB relaxation of a composite function is related to the separate αBB relaxation of both functions. In general this does not commute. We have to refer to the numerical experience in the optimal control literature [PA02] and our own observations in Chapter 6 that the two point multiple shooting approach, including the additional bound on s_1, usually results in a faster overall convergence. Especially at the beginning of the branch-and-bound tree, the αBB underestimator is often worse than these constant bounds.

4.2.2 Adding Additional Multiple Shooting Nodes

Based on Lemma 6, we can start to quantify the quality of the convex relaxation of a direct multiple shooting discretization with more artificial multiple shooting nodes. Therefore, the next step is to analyze the effect of additional multiple shooting nodes. For this purpose, we introduce a new multiple shooting node at a time $t' \in (t_i, t_{i+1})$.
To compare the quality of both convex underestimations, we need a classical result on so-called quasi-monotone functions.

Definition 7 (Quasi-Monotone Function [Wal71]). *The function $f(t,x(t)) : D \subset [t_0, t_N] \times \mathbb{R}^n \mapsto \mathbb{R}^n$ is said to be quasi-monotonically increasing in x, if each component $f_i(t,x)$ is weakly increasing in x_j for $i \neq j$; more exactly, if $\forall i \in \{1, \dots n_x\}$*

$$(t,x) \in D, (t,y) \in D, x \leq y, x_i = y_i \Rightarrow f_i(t,x) \leq f_i(t,y)\,. \tag{4.12}$$

The right-hand side function in our discretized direct optimal control formulation additionally depends on a control parametrization vector q and control values p and we define:

Definition 8 (Quasi-Monotone Function with Discretized Controls and Control Values). *The function $f(t,x(t),\tilde{u}_i(t,q_i),p) : D \subset [t_0, t_N] \times \mathbb{R}^n \mapsto \mathbb{R}^n$ is said to be quasi-monotonically increasing in x, if $\forall (q,p)$ each component $f_i(t,x,\tilde{u}_i(t,q_i),p)$ is weakly increasing in x_j for $i \neq j$; more exactly, if $\forall i \in \{1, \dots n_x\}$, $\forall (q,p)$*

$$(t,x) \in D, (t,y) \in D, x \leq y, x_i = y_i \Rightarrow f_i(t,x,\tilde{u}_i(t,q_i),p) \leq f_i(t,y,\tilde{u}_i(t,q_i),p)\,. \tag{4.13}$$

On each multiple shooting interval $[t_i, t_{i+1}]$, we define a new differential equation with the property that evaluated at t_{i+1} it is equal to the convex underestimator of $x(t_{i+1}; s_i, q_i, p)$. For this purpose, we use the following $\breve{\alpha}/\hat{\alpha}$-augmented right-hand side functions.

Definition 9 ($\breve{\alpha}/\hat{\alpha}$-Augmented RHS Function). *Based on Problem (4.5) with differential equation right-hand side function $f(t,x(t),\tilde{u}_i(t,q_i),p)$, we define on each time interval $[t_i, t_{i+1}] \subseteq [t_0, t_N]$ the $\breve{\alpha}_i$-augmented function $f^{\breve{\alpha}_i} : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \times \mathbb{R}^{n_x} \mapsto \mathbb{R}^n_x$ component-wise $\forall j \in \{1, \dots, n_x\}$ as*

$$\begin{aligned} f_j^{\breve{\alpha}_i}(t,x(t),q,p,s_i) := f_j(t,x(t),\tilde{u}_i(t,q_i),p) + \frac{1}{t_{i+1}-t_i} \Bigg(& \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk} (\overline{s}_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) \\ + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk} (\overline{q}_{i,k} - q_{i,k})(\underline{q}_{i,k} - q_{i,k}) & + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk} (\overline{p}_k - p_k)(\underline{p}_k - p_k) \Bigg) \end{aligned} \tag{4.14}$$

and correspondingly the $\hat{\alpha}$-Augmented function as

$$\begin{aligned} f_j^{\hat{\alpha}_i}(t,x(t),q,p,s_i) := f_j(t,x(t),\tilde{u}_i(t,q_i),p) + \frac{1}{t_{i+1}-t_i} \Bigg(& \sum_{k=1}^{n_x} \hat{\alpha}^{\mathrm{s}}_{ijk} (\overline{s}_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) \\ + \sum_{k=1}^{n_q} \hat{\alpha}^{\mathrm{q}}_{ijk} (\overline{q}_{i,k} - q_{i,k})(\underline{q}_{i,k} - q_{i,k}) & + \sum_{k=1}^{n_p} \hat{\alpha}^{\mathrm{p}}_{ijk} (\overline{p}_k - p_k)(\underline{p}_k - p_k) \Bigg)\,. \end{aligned} \tag{4.15}$$

These augmented functions have the property that they define additional trajectories that, evaluated at t_{i+1}, coincide with the convex/concave relaxation of $x(t_{i+1}; s_i, q_i, p)$. This closes the gap in the state variables of the multiple shooting formulation and allows us to directly compare trajectories in the upcoming proofs instead of a trajectory and a potential jump in the state variables afterwards. We formalize and prove this as the following lemma.

Lemma 10 (Evaluating the $\breve{\alpha}$-Augmented Function)*.* *Let $x(t;s_i,q_i,p)$ be the unique solution to $\dot{x} = f(t,x(t),\tilde{u}_i(t,q_i),p), x(t_i) = s_i$ on $[t_i,t_{i+1}]$. Then the $\breve{\alpha}$-augmented differential equation defined by*

$$x^{\breve{\alpha}_i}(t,x(t),q,p,s_i) = f^{\breve{\alpha}_i}(t,x(t),\tilde{u}_i(t,q_i),p,s_i) \quad x^{\breve{\alpha}_i}(t_0) = x(t_0) \tag{4.16}$$

is unique, exists and is equal to

$$\begin{aligned} x_j^{\breve{\alpha}_i}(t,x(t),q,p,s_i) = x_j(t;s_i,q_i,p) + \frac{t-t_i}{t_{i+1}-t_i}\Bigg(&\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\ &+\sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k)\Bigg). \end{aligned} \tag{4.17}$$

In particular, evaluated at $t = t_{i+1}$, we obtain

$$\begin{aligned} x_j^{\breve{\alpha}_i}(t_{i+1};s_i,q_i,p) = x_j(t_{i+1};s_i,q_i,p) &+ \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\ &+\sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k) \end{aligned} \tag{4.18}$$

which is the αBB-based convex underestimator of $x(t_{i+1};s_i,q_i,p)$.

Proof. Using the integral form of the solution, we can split the integral up into the known solution and an integration of the time independent augmentation term from t_i to t:

$$\begin{aligned} x^{\breve{\alpha}_i}(t,x(t),q,p,s_i) &= s_i + \int_{t_i}^{t} f_j(t,x(t),\tilde{u}_i(t,q_i),p)\mathrm{d}t \\ &\quad + \int_{t_i}^{t} \frac{1}{t_{i+1}-t_i}\Bigg(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\ &\quad +\sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k)\Bigg) \quad (4.19) \\ &= x(t;s_i,q_i,p) + \frac{t-t_i}{t_{i+1}-t_i}\Bigg(\sum_{k=1}^{n_x} \hat{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\ &\quad +\sum_{k=1}^{n_q} \hat{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \hat{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k)\Bigg). \quad (4.20) \end{aligned}$$

Existence and uniqueness follow directly from the corresponding assumption on $x(t;s_i,q_i,p)$. □

Another important aspect for our main results in this section is that quasi-monotonicity of a RHS function is preserved when applying the augmentation. We formalize this as follows.

Lemma 11 (Quasi-Monotonicity of the $\breve{\alpha}$-Augmented Function). *If $f(t,x(t),\tilde{u}_i(t,q_i),p)$ is quasi-monotonically increasing in $x(t)$, then the corresponding $\breve{\alpha}_i$-augmented function as defined in Definition 9 is quasi-monotonically increasing as well.*

Proof. We obtain for $(t,x),(t,y) \in D \subset [t_i,t_{i+1}] \times R^{n_x}, x \leq y, x_j = y_j$:

$$
\begin{aligned}
& f_j^{\breve{\alpha}_i}(t,x(t),\tilde{u}_i(t,q_i),p,s_i) && (4.21)\\
&= f_j(t,x(t),\tilde{u}_i(t,q_i),p) + \frac{1}{t_{i+1}-t_i}\left(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k})\right.\\
&\quad \left. + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k)\right) && (4.22)\\
&\overset{\text{f QM}}{\leq} f_j(t,y(t),\tilde{u}_i(t,q_i),p) + \frac{1}{t_{i+1}-t_i}\left(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k})\right.\\
&\quad \left. + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k)\right) && (4.23)\\
&= f_j^{\breve{\alpha}_i}(t,y,q,p,s_i) && (4.24)
\end{aligned}
$$

$\forall j \in \{1,\ldots,n_x\}$, $\forall (q,p,s_i)$, using the quasi-monotonicity of f and that the augmentation term is less or equal zero $\forall (s_i,q,p)$ by definition and therefore does not change the monotonicity property of f. □

The next step is to compare the $\breve{\alpha}$-augmented function on the full interval $[t_i,t_{i+1}]$ with the locally augmented functions obtained when adding an additional multiple shooting node in between t_i and t_{i+1}. To compare both trajectories, we need a theorem that goes back to [Müller27, Kam32].

Theorem 12 (Quasi-Monotone Function Trajectory Comparison [Wal64]). *Let $f(t,x(t)) : \mathbb{R} \times \mathbb{R}^n \mapsto \mathbb{R}^n$ be quasi-monotonically increasing in $x(t)$. Let $v(t),w(t) : \mathbb{R} \mapsto \mathbb{R}^n$ be continuous in $[t_i,t_{i+1}]$ and differentiable in $(t_i,t_{i+1}]$ and let*

$$v(t_i) \leq w(t_i) \tag{4.25a}$$

$$\dot{v}(t) - f(t,v(t)) \leq \dot{w}(t) - f(t,w(t)) \quad \forall t \in (t_i,t_{i+1}], \tag{4.25b}$$

then

$$v(t) \leq w(t) \quad \forall t \in [t_i,t_{i+1}]. \tag{4.26}$$

With this comparison tool, we can compare the convex under- and overestimations with a different number of multiple shooting nodes.
We start with introducing a common notation used throughout the rest of this chapter to keep track of the original problem DMS^N with N multiple shooting nodes and the new problem DMS^{N+1} with $N+1$ multiple shooting nodes that is derived by adding an additional multiple shooting node at $t' \in [t_i,t_{i+1}]$ to DMS^N. We formalize this as follows:

Definition 13 (Notation for Adding Additional Multiple Shooting Nodes). *Given a multiple shooting based OCP relaxation DMS^N as described in Equation* (4.5) *with N multiple shooting nodes and αBB relaxation parameter vectors $\breve{\alpha}^{\mathrm{s}}$, $\breve{\alpha}^{\mathrm{q}}$ and $\breve{\alpha}^{\mathrm{p}}$, we denote the derived multiple* *shooting relaxation that is obtained by adding a multiple shooting node at $t' \in [t_i, t_{i+1}]$, $i \in \{0, \dots, N-1\}$ by DMS^{N+1}. To distinguish between the different relaxation parameters, $\breve{\alpha}^{\mathrm{s}^\dagger}$, $\breve{\alpha}^{\mathrm{q}^\dagger}$ and $\breve{\alpha}^{\mathrm{p}^\dagger}$ belong to the relaxation of the additional matching condition of DMS^{N+1} at t' and $\breve{\alpha}^{\mathrm{s}^\ddagger}$, $\breve{\alpha}^{\mathrm{q}^\ddagger}$ and $\breve{\alpha}^{\mathrm{p}^\ddagger}$ are the new parameters on the second time interval $[t', t_{i+1}]$. Accordingly, the concave relaxations parameters are $\hat{\alpha}^{\mathrm{s}}$, $\hat{\alpha}^{\mathrm{q}}$ and $\hat{\alpha}^{\mathrm{p}}$ for DMS^N and $\hat{\alpha}^{\mathrm{s}^\dagger}$, $\hat{\alpha}^{\mathrm{q}^\dagger}$, $\hat{\alpha}^{\mathrm{p}^\dagger}$, $\hat{\alpha}^{\mathrm{s}^\ddagger}$, $\hat{\alpha}^{\mathrm{q}^\ddagger}$ and $\hat{\alpha}^{\mathrm{p}^\ddagger}$ for DMS^{N+1}.*

To simplify the following theorem itself and the corresponding proof, we include rather strong assumptions on the values of α first and proceed afterwards to analyze how we can fulfill or even relax them.

Theorem 14 (Adding Multiple Shooting Nodes). *Given a problem as defined in Equation* (4.5) *with a quasi-monotone right-hand side function $f(t, x(t), \tilde{u}_i(t, q_i), p)$ on $[t_i, t_{i+1}]$ w.r.t. $x(t)$, $\forall (q, p)$, $\forall i = 0, \dots, N$ adding an additional multiple shooting node at $t' \in (t_i, t_{i+1})$ leads to a new convex relaxation with optimum Φ^{N+1} and it holds true that $\Phi^{N+1} \geq \Phi^N$ if*

$$\breve{\alpha}^{\mathrm{s}^\dagger}_{ijk_s} \leq \breve{\alpha}^{\mathrm{s}}_{ijk_s} \qquad \breve{\alpha}^{\mathrm{s}^\ddagger}_{ijk_s} = 0 \tag{4.27a}$$
$$\breve{\alpha}^{\mathrm{q}^\dagger}_{ijk_q} \leq \breve{\alpha}^{\mathrm{q}}_{ijk_q} \qquad \breve{\alpha}^{\mathrm{q}^\ddagger}_{ijk_q} \leq \breve{\alpha}^{\mathrm{q}}_{ijk_q} \tag{4.27b}$$
$$\breve{\alpha}^{\mathrm{p}^\dagger}_{ijk_p} \leq \breve{\alpha}^{\mathrm{p}}_{ijk_p} \qquad \breve{\alpha}^{\mathrm{p}^\ddagger}_{ijk_p} \leq \breve{\alpha}^{\mathrm{p}}_{ijk_p} \tag{4.27c}$$

and

$$\hat{\alpha}^{\mathrm{s}^\dagger}_{ijk_s} \leq \hat{\alpha}^{\mathrm{s}}_{ijk_s} \qquad \hat{\alpha}^{\mathrm{s}^\ddagger}_{ijk_s} = 0 \tag{4.28a}$$
$$\hat{\alpha}^{\mathrm{q}^\dagger}_{ijk_q} \leq \hat{\alpha}^{\mathrm{q}}_{ijk_q} \qquad \hat{\alpha}^{\mathrm{q}^\ddagger}_{ijk_q} \leq \hat{\alpha}^{\mathrm{q}}_{ijk_q} \tag{4.28b}$$
$$\hat{\alpha}^{\mathrm{p}^\dagger}_{ijk_p} \leq \hat{\alpha}^{\mathrm{p}}_{ijk_p} \qquad \hat{\alpha}^{\mathrm{p}^\ddagger}_{ijk_p} \leq \hat{\alpha}^{\mathrm{p}}_{ijk_p} \tag{4.28c}$$

$\forall i \in \{0, \dots, N-1\}$, $\forall j \in \{1, \dots, n_x\}$, $\forall k^s \in \{1, \dots, n_x\}$, $\forall k_q \in \{1, \dots, n_q\}$, $\forall k_p \in \{1, \dots, n_p\}$.

In the following proof, we make use of the integral form of different solutions of the underlying ODE. To keep track of the notation of the unique trajectories that differ in the initial values only, we include the initial value of the integral at its lower bound directly below the integral sign, e.g.

$$x_j(t_{i+1}; s_i, q_i, p) = s_{i,j} + \int\limits_{\substack{t_i \\ x_j(t_i) = s_{i,j}}}^{t_i+1} f_j(t, x(t), \tilde{u}_i(t, q_i), p) \mathrm{d}t \, . \tag{4.29}$$

The general idea is to compare both relaxations through their feasible trajectories using the α-augmented functions introduced in Lemma 10, as shown in Figure 4.1. In preparation of the main part of this proof, where we apply Theorem 12, we derive the two necessary underestimations in Assumptions 4.25a and 4.25b now.

First, we solve the new relaxed matching conditions at t' from Equation (4.5b) for s'_j and use the left column of the Assumptions 4.27 to obtain:

$$\begin{aligned}
s'_j &\geq x_j(t';s_i,q_i,p) + \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\dagger}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\dagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\dagger}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k) \qquad (4.30a) \\
&\geq x_j(t';s_i,q_i,p) + \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k) \qquad (4.30b) \\
&= x^{\breve{\alpha}}_j(t';s_i,q_i,p) \quad \forall j \in \{1,\dots,n_x\}\,. \qquad (4.30c)
\end{aligned}$$

The second preparation is to verify that

$$\begin{aligned}
& f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i}\left(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\ddagger}_{ijk}(\overline{s}'_k-s'_k)(\underline{s}'_k-s'_k)\right. \\
&\left. + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\ddagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\ddagger}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k)\right) \qquad (4.31a) \\
\geq\ & f_j(t,y(t),q,p) + \frac{1}{t_{i+1}-t_i}\left(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k})\right. \\
&\left. + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k)\right), \qquad (4.31b)
\end{aligned}$$

which holds true due to Assumption 4.27a. Especially when $s_{i,k} = \underline{s}_{i,k} = \overline{s}_{i,k}$ is fixed and the interval width $\mathrm{w}([\underline{s}'_j,\overline{s}'_j]) > 0$, the necessity for this very restrictive assumption becomes obvious. Nevertheless, we are able to relax it in the following sections.
To compare the feasible sets of both relaxations, we now estimate $s_{i+1,j}$ from below, once more, by solving the relaxed matching conditions from Equation (4.5b), this time for $s_{i+1,j}$, and we obtain the following proof of Theorem 14.

Proof.

$$\begin{aligned}
s_{i+1,j} &\geq x_j(t_{i+1};s',q_i,p) + \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\ddagger}_{ijk}(\overline{s}'_k-s'_k)(\underline{s}'_k-s'_k) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\ddagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\ddagger}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k) \qquad (4.32a) \\
&= s'_j + \int_{\substack{t' \\ x_j(t')=s'_j}}^{t_{i+1}} \left(f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i}\left(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\ddagger}_{ijk}(\overline{s}'_k-s'_k)(\underline{s}'_k-s'_k)\right.\right. \\
&\quad \left.\left. + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\ddagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\ddagger}_{ijk}(\overline{p}_k-p_k)(\underline{p}_k-p_k)\right)\right)\mathrm{d}t \qquad (4.32b)
\end{aligned}$$

$$
\begin{aligned}
&\geq x_j(t';s_i,q_i,p) + \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\dagger}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\dagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\dagger}_{ijk}(\overline{p}_{k}-p_{k})(\underline{p}_{k}-p_{k}) \\
&\quad + \int\limits_{\substack{t' \\ x_j(t')=s'_j}}^{t_{i+1}} \Bigg(f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i} \Bigg(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\ddagger}_{ijk}(\overline{s}'_{k}-s'_{k})(\underline{s}'_{k}-s'_{k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\ddagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\ddagger}_{ijk}(\overline{p}_{k}-p_{k})(\underline{p}_{k}-p_{k}) \Bigg)\Bigg) \mathrm{d}t \qquad (4.32c) \\
&= s_{i,j} + \int\limits_{\substack{t_i \\ x_j(t_i)=s_{i,j}}}^{t'} \Bigg(f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i} \Bigg(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\dagger}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\dagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\dagger}_{ijk}(\overline{p}_{k}-p_{k})(\underline{p}_{k}-p_{k}) \Bigg)\Bigg) \mathrm{d}t \\
&\quad + \int\limits_{\substack{t' \\ x_j(t')=s'_j}}^{t_{i+1}} \Bigg(f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i} \Bigg(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\ddagger}_{ijk}(\overline{s}'_{k}-s'_{k})(\underline{s}'_{k}-s'_{k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\ddagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\ddagger}_{ijk}(\overline{p}_{k}-p_{k})(\underline{p}_{k}-p_{k}) \Bigg)\Bigg) \mathrm{d}t \qquad (4.32d) \\
&\geq s_{i,j} + \int\limits_{\substack{t_i \\ x_j(t_i)=s_{i,j}}}^{t'} \Bigg(f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i} \Bigg(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\dagger}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\dagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\dagger}_{ijk}(\overline{p}_{k}-p_{k})(\underline{p}_{k}-p_{k}) \Bigg)\Bigg) \mathrm{d}t \\
&\quad + \int\limits_{\substack{t' \\ x_j(t')=x_j^{\breve{\alpha}}(t';s_i,q_i,p)}}^{t_{i+1}} \Bigg(f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i} \Bigg(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}^\ddagger}_{ijk}(\overline{s}'_{k}-s'_{k})(\underline{s}'_{k}-s'_{k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}^\ddagger}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}^\ddagger}_{ijk}(\overline{p}_{k}-p_{k})(\underline{p}_{k}-p_{k}) \Bigg)\Bigg) \mathrm{d}t \qquad (4.32e) \\
&\geq s_{i,j} + \int\limits_{\substack{t_i \\ x_j(t_i)=s_{i,j}}}^{t'} \Bigg(f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i} \Bigg(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_{k}-p_{k})(\underline{p}_{k}-p_{k}) \Bigg)\Bigg) \mathrm{d}t \\
&\quad + \int\limits_{\substack{t' \\ x_j(t')=x_j^{\breve{\alpha}}(t';s_i,q_i,p)}}^{t_{i+1}} \Bigg(f_j(t,x(t),q,p) + \frac{1}{t_{i+1}-t_i} \Bigg(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk}(\overline{s}_{i,k}-s_{i,k})(\underline{s}_{i,k}-s_{i,k}) \\
&\quad + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k}-q_{i,k})(\underline{q}_{i,k}-q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_{k}-p_{k})(\underline{p}_{k}-p_{k}) \Bigg)\Bigg) \mathrm{d}t \qquad (4.32f)
\end{aligned}
$$

$$= s_{i,j} + \int_{\substack{t_i \\ x_j(t_i)=s_{i,j}}}^{t_{i+1}} \left(f_j(t, x(t), q, p) + \frac{1}{t_{i+1} - t_i} \left(\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk} (\overline{s}_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) \right.\right.$$
$$\left.\left. + \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk} (\overline{q}_{i,k} - q_{i,k})(\underline{q}_{i,k} - q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk} (\overline{p}_k - p_k)(\underline{p}_k - p_k) \right) \right) \mathrm{dt} \tag{4.32g}$$

$$= x(t_{i+1}; s', q_i, p) + \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{ijk} (\overline{s}_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k})$$
$$+ \sum_{k=1}^{n_q} \breve{\alpha}^{\mathrm{q}}_{ijk} (\overline{q}_{i,k} - q_{i,k})(\underline{q}_{i,k} - q_{i,k}) + \sum_{k=1}^{n_p} \breve{\alpha}^{\mathrm{p}}_{ijk} (\overline{p}_k - p_k)(\underline{p}_k - p_k), \tag{4.32h}$$

where we apply Lemma 10 from Equation (4.32a) to (4.32b). The next step is to replace s'_j by its lower bound from the additional relaxed matching condition at t' leading to Equation (4.32c). Once more applying Lemma 10 results in Equation (4.32d).

In the next step, from Equation (4.32d) to Equation (4.32e), the integral initial values become important, because we have to estimate the gap between the trajectories. Using the quasi-monotonicity of the augmented function that was shown in Lemma 11 together with Equations (4.30) and (4.31), we are able to apply Theorem 12 and estimate once more from below by decreasing the initial value of the second integral from s_j to $x_j^{\breve{\alpha}}(t'; s_i, q_i, p)$.

Afterwards, we use Assumption (4.27) to underestimate both integrals, because the augmentation terms are negative by definition, resulting in Equation (4.32f). The last step is to combine both integrals that act on the same trajectory now and finally using Lemma 10 once more to obtain the lower relaxed matching condition of DMS^N on $s_{i+1,j}$.

Accordingly, with reversed estimates, we show that Inequality (4.5d), as an upper bound on $s_{i+1,j}$, is under the assumption a tighter bound in problem DMS^{N+1}. Therefore, the feasible set of DMS^{N+1} is a subset of DMS^N and it holds true that $\Phi^{N+1} \geq \Phi^N$. □

We continue this Chapter by showing under which circumstances the assumptions of Theorem 14 are fulfilled.

4.3 Influence of Additional Multiple Shooting Nodes on α

This section is dedicated to analyzing and relaxing the assumptions of Theorem 14 in Equation (4.27). Our first focus is the Assumption (4.27a).

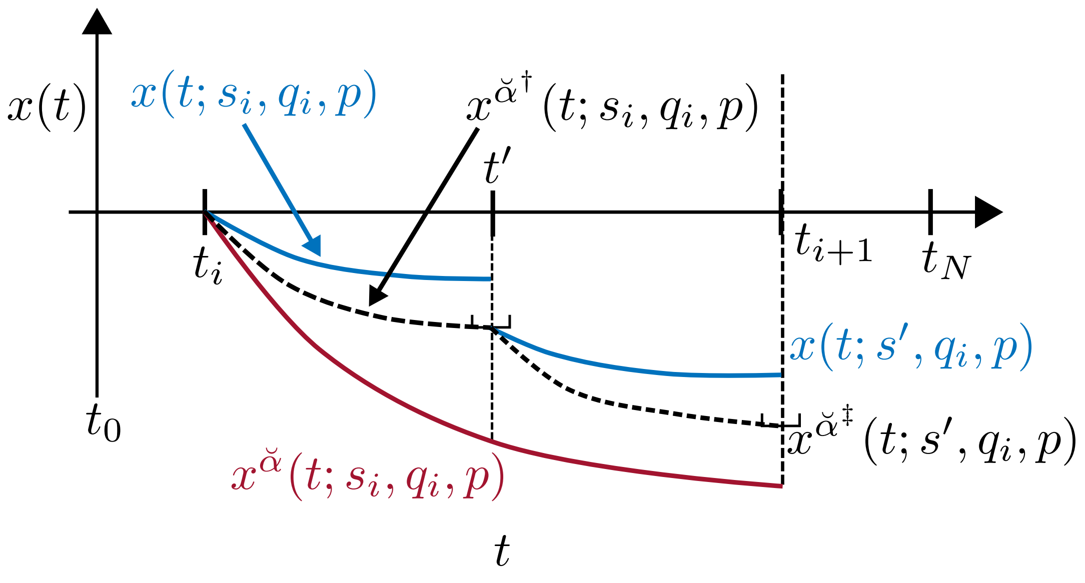

$x(t)$
$x(t; s_i, q_i, p)$
$x^{\breve\alpha^\dagger}(t; s_i, q_i, p)$
t'
t_i
t_{i+1}
t_N
$x(t; s', q_i, p)$
t_0
$x^{\breve\alpha^\ddagger}(t; s', q_i, p)$
$x^{\breve\alpha}(t; s_i, q_i, p)$
t

Figure 4.1: An illustration of the idea to the proof of Theorem 14.

4.3.1 Rate of Convergence of the Over- and Underestimations With Respect to the Multiple Shooting Variables

Solving both additional relaxed matching conditions at t' for $x_j(t'; t_i, s_i, q_i, p) - s'_j$, we obtain componentwise $\forall j \in \{1, \dots, n_x\}$ the under- respectively overestimation:

$$\begin{aligned} s'_j \geq x_j(t'; t_i, s_i, q_i, p) &+ \sum_{k=1}^{n_x} \breve\alpha^{s^\dagger}_{ijk}(\bar s_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) \\ &+ \sum_{k=1}^{n_q} \breve\alpha^{q^\dagger}_{ijk}(\bar q_{i,k} - q_{i,k})(\underline{q}_{i,k} - q_{i,k}) + \sum_{k=1}^{n_p} \breve\alpha^{p^\dagger}_{ijk}(\bar p_k - p_k)(\underline{p}_k - p_k) \end{aligned} \tag{4.33a}$$

$$\begin{aligned} s'_j \leq x_j(t'; t_i, s_i, q_i, p) &- \sum_{k=1}^{n_x} \hat\alpha^{s^\dagger}_{ijk}(\bar s_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) \\ &- \sum_{k=1}^{n_q} \hat\alpha^{q^\dagger}_{ijk}(\bar q_{i,k} - q_{i,k})(\underline{q}_{i,k} - q_{i,k}) - \sum_{k=1}^{n_p} \hat\alpha^{p^\dagger}_{ijk}(\bar p_k - p_k)(\underline{p}_k - p_k)\,. \end{aligned} \tag{4.33b}$$

The reason for the assumption $\alpha = 0$ is the integrand

$$\sum_{k=1}^{n_x} \breve\alpha^{s^\ddagger}_{ijk}(\bar s'_k - s'_k)(\underline{s}'_k - s'_k) \tag{4.34}$$

in Equation (4.32e). Using Equation (4.33), we can derive a lower bound.

$$0 \geq \sum_{k=1}^{n_x} \breve\alpha^{s^\ddagger}_{ijk}(\bar s'_k - s'_k)(\underline{s}'_k - s'_k) \tag{4.35a}$$

$$\geq -\sum_{k=1}^{n_x} \breve{\alpha}^{s^\ddagger}_{ijk} \frac{(\overline{s}'_k - \underline{s}'_k)^2}{4} \tag{4.35b}$$

$$\begin{aligned}
= -\sum_{k=1}^{n_x} \frac{\breve{\alpha}^{s^\ddagger}_{ijk}}{4} \Bigg(& x_k(t'; t_i, s_i, q_i, p) + \sum_{l=1}^{n_x} \hat{\alpha}^{s^\dagger}_{ikl}(\overline{s}_{i,l} - s_{i,l})(\underline{s}_{i,l} - s_{i,l}) \\
& + \sum_{l=1}^{n_q} \hat{\alpha}^{q^\dagger}_{ikl}(\overline{q}_{i,l} - q_{i,l})(\underline{q}_{i,l} - q_{i,l}) + \sum_{l=1}^{n_p} \hat{\alpha}^{p^\dagger}_{ikl}(\overline{p}_l - p_l)(\underline{p}_l - p_l) \\
& - x_k(t'; t_i, s_i, q_i, p) + \sum_{l=1}^{n_x} \breve{\alpha}^{s^\dagger}_{ikl}(\overline{s}_{i,l} - s_{i,l})(\underline{s}_{i,l} - s_{i,l}) \\
& + \sum_{l=1}^{n_q} \breve{\alpha}^{q^\dagger}_{ikl}(\overline{q}_{i,l} - q_{i,l})(\underline{q}_{i,l} - q_{i,l}) + \sum_{l=1}^{n_p} \breve{\alpha}^{p^\dagger}_{ikl}(\overline{p}_l - p_l)(\underline{p}_l - p_l) \Bigg)^2
\end{aligned} \tag{4.35c}$$

$$\begin{aligned}
= -\sum_{k=1}^{n_x} \frac{\breve{\alpha}^{s^\ddagger}_{ijk}}{4} \Bigg(& \sum_{l=1}^{n_x} (\breve{\alpha}^{s^\dagger}_{ikl} + \hat{\alpha}^{s^\dagger}_{ikl})(\overline{s}_{i,l} - s_{i,l})(\underline{s}_{i,l} - s_{i,l}) \\
& + \sum_{l=1}^{n_q} (\breve{\alpha}^{q^\dagger}_{ikl} + \hat{\alpha}^{q^\dagger}_{ikl})(\overline{q}_{i,l} - q_{i,l})(\underline{q}_{i,l} - q_{i,l}) + \sum_{l=1}^{n_p} (\breve{\alpha}^{p^\dagger}_{ikl} + \hat{\alpha}^{p^\dagger}_{ikl})(\overline{p}_l - p_l)(\underline{p}_l - p_l) \Bigg)^2
\end{aligned} \tag{4.35d}$$

$$\begin{aligned}
\geq -\sum_{k=1}^{n_x} \frac{\breve{\alpha}^{s^\ddagger}_{ijk}}{16} \Bigg(& \sum_{l=1}^{n_x} (\breve{\alpha}^{s^\dagger}_{ikl} + \hat{\alpha}^{s^\dagger}_{ikl})(\overline{s}_{i,l} - \underline{s}_{i,l})^2 \\
& + \sum_{l=1}^{n_q} (\breve{\alpha}^{q^\dagger}_{ikl} + \hat{\alpha}^{q^\dagger}_{ikl})(\overline{q}_{i,l} - \underline{q}_{i,l})^2 + \sum_{l=1}^{n_p} (\breve{\alpha}^{p^\dagger}_{ikl} + \hat{\alpha}^{p^\dagger}_{ikl})(\overline{p}_l - \underline{p}_l)^2 \Bigg)^2 .
\end{aligned} \tag{4.35e}$$

Interpreting the convex underestimation as a sequence in $w([\underline{s}, \overline{s}]) \to 0$, $w([\underline{q}, \overline{q}]) \to 0$ and $w([\underline{p}, \overline{p}]) \to 0$ during the branch-and-bound that is bounded from above by zero, we have derived an additional lower bound that depends only on fourth degree monomials in the variable interval widths, compared to the usual quadratic convergence rate of the αBB underestimator. Therefore, although this term may not be equal zero in general it vanishes much faster then the convex relaxations with respect to the discretized controls and control values.

We now generalize the observation from Inequality (4.35) to the case of arbitrary many multiple shooting nodes by applying this technique recursively using the corresponding relaxed matching conditions on each node. This way, we are able to under- and overestimate the convex relaxations of the direct multiple-shooting-based approach with respect to $s_1, \ldots, s_N$ by replacing all dependencies on the interval width of these terms with monomials in the interval widths of $\mathrm{w}(\underline{s}_0, \overline{s}_0)$, $\mathrm{w}(\underline{q}, \overline{q})$ and $\mathrm{w}(\underline{p}, \overline{p})$ of at least fourth degree.

This is stated in the following theorem.

Theorem 15 (Rate of Convergence of the Over- and Underestimations With Respect to the Multiple Shooting Variables). *Given Problem* (4.5)*, $\forall i \in \{1, \ldots, N\}$ and $\forall j \in \{1, \ldots, n_x\}$, $\exists c_1, c_2 \in \mathbb{R}, c_1 > 0, c_2 > 0$, such that*

$$0 \geq \sum_{k=1}^{n_x} \breve{\alpha}^{s}_{ijk}(\overline{s}_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) \tag{4.36a}$$

$$
\begin{aligned}
&\geq -c_1 \sum_{k=1}^{n_x} (\bar{s}_{i,k} - \underline{s}_{i,k})^2 && \text{(4.36b)}\\
&\geq -c_2 \left(\left(\sum_{l=1}^{n_x} \bar{s}_{0,l} - \underline{s}_{0,l} \right)^{2^{i+1}} + \sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_q} \bar{q}_{k,l} - \underline{q}_{k,l} \right)^4 + \sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_p} \bar{p}_l - \underline{p}_l \right)^4 \right.\\
&\quad + 2 \left(\sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_x} \bar{s}_{0,l} - \underline{s}_{0,l} \right)^{2^k} \left(\sum_{l=1}^{n_q} \bar{q}_{k,l} - \underline{q}_{k,l} \right)^2 \right.\\
&\quad \left.\left. + \sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_x} \bar{s}_{0,l} - \underline{s}_{0,l} \right)^{2^i} \left(\sum_{l=1}^{n_p} \bar{p}_l - \underline{p}_l \right)^2 + \sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_q} \bar{q}_{k,l} - \underline{q}_{k,l} \right)^2 \left(\sum_{l=1}^{n_p} \bar{p}_l - \underline{p}_l \right)^2 \right) \right) && \text{(4.36c)}
\end{aligned}
$$

and correspondingly

$$
\begin{aligned}
0 &\geq \sum_{k=1}^{n_x} \hat{\alpha}^{\mathrm{s}}_{ijk} (\bar{s}_{i,k} - s_{i,k})(\underline{s}_{i,k} - s_{i,k}) && \text{(4.37a)}\\
&\geq -c_1 \sum_{k=1}^{n_x} (\bar{s}_{i,k} - \underline{s}_{i,k})^2 && \text{(4.37b)}\\
&\geq -c_2 \left(\left(\sum_{l=1}^{n_x} \bar{s}_{0,l} - \underline{s}_{0,l} \right)^{2^{i+1}} + \sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_q} \bar{q}_{k,l} - \underline{q}_{k,l} \right)^4 + \sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_p} \bar{p}_l - \underline{p}_l \right)^4 \right.\\
&\quad + 2 \left(\sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_x} \bar{s}_{0,l} - \underline{s}_{0,l} \right)^{2^k} \left(\sum_{l=1}^{n_q} \bar{q}_{k,l} - \underline{q}_{k,l} \right)^2 \right.\\
&\quad \left.\left. + \sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_x} \bar{s}_{0,l} - \underline{s}_{0,l} \right)^{2^i} \left(\sum_{l=1}^{n_p} \bar{p}_l - \underline{p}_l \right)^2 + \sum_{k=0}^{i-1} \left(\sum_{l=1}^{n_q} \bar{q}_{k,l} - \underline{q}_{k,l} \right)^2 \left(\sum_{l=1}^{n_p} \bar{p}_l - \underline{p}_l \right)^2 \right) \right). && \text{(4.37c)}
\end{aligned}
$$

Proof. By induction over i, we obtain for

i=1: Using Equation (4.35) and replacing $s_{i,k}$ with $s_{0,k}$ and s'_k with $s_{1,k}$, we over- and underestimate $\forall j \in \{1, \dots, n_x\}$ as follows:

$$
\begin{aligned}
0 &\geq \sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{1jk} (\bar{s}_{1,k} - s_{1,k})(\underline{s}_{1,k} - s_{1,k}) && \text{(4.38a)}\\
&\geq -\sum_{k=1}^{n_x} \breve{\alpha}^{\mathrm{s}}_{1jk} \frac{(\bar{s}_{1,k} - \underline{s}_{1,k})^2}{4} && \text{(4.38b)}\\
&\geq -c_1 \sum_{k=1}^{n_x} (\bar{s}_{1,k} - \underline{s}_{1,k})^2 && \text{(4.38c)}\\
&\geq -\frac{c_1}{4} \sum_{k=1}^{n_x} \left(\sum_{l=1}^{n_x} (\breve{\alpha}^{\mathrm{s}}_{0kl} + \hat{\alpha}^{\mathrm{s}}_{0kl})(\bar{s}_{0,l} - \underline{s}_{0,l})^2 \right.\\
&\quad \left. + \sum_{l=1}^{n_q} (\breve{\alpha}^{\mathrm{q}}_{0kl} + \hat{\alpha}^{\mathrm{q}}_{0kl})(\bar{q}_{0,l} - \underline{q}_{0,l})^2 + \sum_{l=1}^{n_p} (\breve{\alpha}^{\mathrm{p}}_{0kl} + \hat{\alpha}^{\mathrm{p}}_{0kl})(\bar{p}_l - \underline{p}_l)^2 \right)^2 && \text{(4.38d)}\\
&\geq -c_2 \left(\left(\sum_{l=1}^{n_x} \bar{s}_{0,l} - \underline{s}_{0,l} \right)^{2^{i+1}} + \sum_{k=0}^{0} \left(\sum_{l=1}^{n_q} \bar{q}_{k,l} - \underline{q}_{k,l} \right)^4 + \sum_{k=0}^{0} \left(\sum_{l=1}^{n_p} \bar{p}_l - \underline{p}_l \right)^4 \right. && \text{(4.38e)}
\end{aligned}
$$

$$+2\left(\sum_{k=0}^{0}\left(\sum_{l=1}^{n_x}\bar{s}_{0,l}-\underline{s}_{0,l}\right)^{2^k}\left(\sum_{l=1}^{n_q}\bar{q}_{k,l}-\underline{q}_{k,l}\right)^2\right. \tag{4.38f}$$

$$\left.+\sum_{k=0}^{0}\left(\sum_{l=1}^{n_x}\bar{s}_{0,l}-\underline{s}_{0,l}\right)^{2^i}\left(\sum_{l=1}^{n_p}\bar{p}_l-\underline{p}_l\right)^2+\sum_{k=0}^{0}\left(\sum_{l=1}^{n_q}\bar{q}_{k,l}-\underline{q}_{k,l}\right)^2\left(\sum_{l=1}^{n_p}\bar{p}_l-\underline{p}_l\right)^2\right) \tag{4.38g}$$

with

$$c_1 \geq \frac{1}{4}\max_{\substack{j=1,\dots,n_x\\ k=1,\dots,n_x}}\left(\breve{\alpha}^{\mathrm{s}}_{1jk}\right) \tag{4.39}$$

and

$$c_2 \geq \frac{c_1 n_x}{4}\left(\max_{\substack{k=1,\dots,n_x\\ l_s=1,\dots,n_x\\ l_q=1,\dots,n_q\\ l_p=1,\dots,n_p}}\left((\breve{\alpha}^{\mathrm{s}}_{0kl_s}+\hat{\alpha}^{\mathrm{s}}_{0kl_s}),(\breve{\alpha}^{\mathrm{q}}_{0kl_q}+\hat{\alpha}^{\mathrm{q}}_{0kl_q}),(\breve{\alpha}^{\mathrm{p}}_{0kl_p}+\hat{\alpha}^{\mathrm{p}}_{0kl_p})\right)\right)^2. \tag{4.40}$$

$i \to i+1$: Using Equation (4.35) once more and this time replacing only s'_k with $s_{i+1,k}$, we over- and underestimate $\forall j = 1,\dots,n_x$ as follows:

$$0 \geq \sum_{k=1}^{n_x}\breve{\alpha}^{\mathrm{s}}_{i+1,jk}(\bar{s}_{i+1,k}-s_{i+1,k})(\underline{s}_{i+1,k}-s_{i+1,k}) \tag{4.41a}$$

$$\geq -\sum_{k=1}^{n_x}\breve{\alpha}^{\mathrm{s}}_{i+1,jk}\frac{(\bar{s}_{i+1,k}-\underline{s}_{i+1,k})^2}{4} \tag{4.41b}$$

$$\geq -c'_1\sum_{k=1}^{n_x}(\bar{s}_{i+1,k}-\underline{s}_{i+1,k})^2 \tag{4.41c}$$

$$\geq -\frac{c'_1}{4}\sum_{k=1}^{n_x}\left(\sum_{l=1}^{n_x}(\breve{\alpha}^{\mathrm{s}}_{ikl}+\hat{\alpha}^{\mathrm{s}}_{ikl})(\bar{s}_{i,l}-\underline{s}_{i,l})^2\right.$$
$$\left.+\sum_{l=1}^{n_q}(\breve{\alpha}^{\mathrm{q}}_{ikl}+\hat{\alpha}^{\mathrm{q}}_{ikl})(\bar{q}_{i,l}-\underline{q}_{i,l})^2+\sum_{l=1}^{n_p}(\breve{\alpha}^{\mathrm{p}}_{ikl}+\hat{\alpha}^{\mathrm{p}}_{ikl})(\bar{p}_l-\underline{p}_l)^2\right)^2 \tag{4.41d}$$

$$\geq -\frac{c'_1}{4}\sum_{k=1}^{n_x}\left(\sum_{l=1}^{n_x}(\breve{\alpha}^{\mathrm{s}}_{ikl}+\hat{\alpha}^{\mathrm{s}}_{ikl})\frac{c_2}{c_1}\left(\left(\sum_{l=1}^{n_x}\bar{s}_{0,l}-\underline{s}_{0,l}\right)^{2^{i+1}}+\sum_{k=0}^{i-1}\left(\sum_{l=1}^{n_q}\bar{q}_{k,l}-\underline{q}_{k,l}\right)^4\right.\right.$$
$$+\sum_{k=0}^{i-1}\left(\sum_{l=1}^{n_p}\bar{p}_l-\underline{p}_l\right)^4+2\left(\sum_{k=0}^{i-1}\left(\sum_{l=1}^{n_x}\bar{s}_{0,l}-\underline{s}_{0,l}\right)^{2^k}\left(\sum_{l=1}^{n_q}\bar{q}_{k,l}-\underline{q}_{k,l}\right)^2\right.$$
$$\left.\left.+\sum_{k=0}^{i-1}\left(\sum_{l=1}^{n_x}\bar{s}_{0,l}-\underline{s}_{0,l}\right)^{2^i}\left(\sum_{l=1}^{n_p}\bar{p}_l-\underline{p}_l\right)^2+\sum_{k=0}^{i-1}\left(\sum_{l=1}^{n_q}\bar{q}_{k,l}-\underline{q}_{k,l}\right)^2\left(\sum_{l=1}^{n_p}\bar{p}_l-\underline{p}_l\right)^2\right)\right)$$
$$\left.+\sum_{l=1}^{n_q}(\breve{\alpha}^{\mathrm{q}}_{ikl}+\hat{\alpha}^{\mathrm{q}}_{ikl})(\bar{q}_{i,l}-\underline{q}_{i,l})^2+\sum_{l=1}^{n_p}(\breve{\alpha}^{\mathrm{p}}_{ikl}+\hat{\alpha}^{\mathrm{p}}_{ikl})(\bar{p}_l-\underline{p}_l)^2\right)^2 \tag{4.41e}$$

$$
\begin{aligned}
&\geq -c_2'\left(\sum_{l=1}^{n_x}\bar{s}_{0,l}-\underline{s}_{0,l}\right)^{2^{i+2}}+\sum_{k=0}^{i}\left(\sum_{l=1}^{n_q}\bar{q}_{k,l}-\underline{q}_{k,l}\right)^{4}+\sum_{k=0}^{i}\left(\sum_{l=1}^{n_p}\bar{p}_{l}-\underline{p}_{l}\right)^{4}\\
&\quad+2\left(\sum_{k=0}^{i}\left(\sum_{l=1}^{n_x}\bar{s}_{0,l}-\underline{s}_{0,l}\right)^{2^{k}}\left(\sum_{l=1}^{n_q}\bar{q}_{k,l}-\underline{q}_{k,l}\right)^{2}\right.\\
&\quad\left.+\sum_{k=0}^{i}\left(\sum_{l=1}^{n_x}\bar{s}_{0,l}-\underline{s}_{0,l}\right)^{2^{i+1}}\left(\sum_{l=1}^{n_p}\bar{p}_{l}-\underline{p}_{l}\right)^{2}+\sum_{k=0}^{i}\left(\sum_{l=1}^{n_q}\bar{q}_{k,l}-\underline{q}_{k,l}\right)^{2}\left(\sum_{l=1}^{n_p}\bar{p}_{l}-\underline{p}_{l}\right)^{2}\right)
\end{aligned}
\tag{4.41f}
$$

with

$$
c_1' \geq \frac{1}{4}\max_{\substack{j=1,\dots,n_x\\ k=1,\dots,n_x}}\left(\breve{\alpha}^{\mathrm{s}}_{i+1,jk}\right) \tag{4.42}
$$

and

$$
c_2' \geq \frac{c_1' n_x}{4}\left(\max_{\substack{k=1,\dots,n_x\\ l_s=1,\dots,n_x\\ l_q=1,\dots,n_q\\ l_p=1,\dots,n_p}}\left(\frac{c_2}{c_1}(\breve{\alpha}^{\mathrm{s}}_{ikl_s}+\hat{\alpha}^{\mathrm{s}}_{ikl_s}),(\breve{\alpha}^{\mathrm{q}}_{ikl_q}+\hat{\alpha}^{\mathrm{q}}_{ikl_q}),(\breve{\alpha}^{\mathrm{p}}_{ikl_p}+\hat{\alpha}^{\mathrm{p}}_{ikl_p})\right)\right)^{2}, \tag{4.43}
$$

using the induction hypothesis from Equation (4.41d) to (4.41e).
And accordingly for the overestimation of the concave αBB term w.r.t. the multiple shooting variables. □

Therefore, for $\mathrm{w}([\underline{s}_0,\bar{s}_0]) \to 0$, $\mathrm{w}([\underline{q},\bar{q}]) \to 0$ and $\mathrm{w}([\underline{p},\bar{p}]) \to 0$, the rate of convergence of the αBB over- and underestimator with respect to $s_1,\dots,s_N$ is two orders higher compared to the quadratic convergence rate of the corresponding terms in q and p.
This gives a theoretical explanation why branching solely on q and p, as described in Section 4.4.1, is not only a valid choice that maintains convergence, but works very efficiently in practice. Furthermore, for problems with fixed or partially fixed initial values, the monomials including $(\bar{s}_0-\underline{s}_0)$ become zero as well. This motivates and justifies the reduced space heuristic described later in Section 4.4.3.
This convergence behavior, resulting in a rapid bound tightening with respect to the multiple shooting variables in the lifted variable space, shows similarities with the observations made in [AD10] for the convergence of a lifted Newton approach.
In the next section, we take a closer look at the second-order sensitivities and the effect of adding additional multiple shooting nodes on the bounds of the solution of the variational differential equations.

4.3.2 Influence of the Constant Initial Value of the Variational Differential Equations

The next step is to exploit the fact that the initial value of the second-order interval sensitivities $H(t;s_i,q_i,p)$ is $H(t_i;s_i,q_i,p)=[0,0]$. This allows us to prove that the left columns of Assumption (4.27) and Assumption (4.28) are automatically fulfilled under some assumptions on the second-order sensitivities in theory and almost every time for practical applications.
To ease the notation in the following theorem and proof, we use an indexed tensor notation for $H_{jkl}(t;s_i,q_i,p)$ instead of the formulation given in Equation (2.14) with $j \in \{1,\dots,n_x\}$

being the dimension of the state space and therefore, for fixed j, $H_{jkl}(t;s_i,q_i,p)$ is the interval Hessian of $x_j(t)$ with components $k,l \in \{1,\dots,(n_x+n_q+n_p)\}$. Furthermore, we define the k-th radius of the corresponding Gershgorin disc of $H_j(t;s_i,q_i,p))$ as $R_k(H_j(t;s_i,q_i,p))$. Finally, as in Section 4.2 and formalized in Definition 13, we indicate new variables after introducing a multiple shooting node $t' \in [t_i,t_{i+1}]$ on $[t_i,t']$ with † and on $[t',t_{i+1}]$ with ‡ to distinguish them from the original variables on $[t_i,t_{i+1}]$.
Using Gershgorin's theorem to underestimate the eigenvalues of the interval Hessians, the following theorem requires an additional assumption on the diagonal entries. Namely that if the lower bound of a diagonal entry already has a positive sign, the difference in the radius for different initial values has to be greater than the difference in the diagonal entry itself. Likewise, for the overestimation of the eigenvalues that if the upper bound of a diagonal entry on one of the interval Hessians is already negative, the difference in the corresponding radius has to exceed the difference in the diagonal entry itself.
We state this assumption this as follows.

Assumption 16 (Difference of Diagonal and Radius). *Given Assumption 4 and a problem as defined in Equation* (4.5)*, adding an additional multiple shooting node at $t' \in (t_i,t_{i+1})$ leads to a new convex relaxation. We assume for the bounds on the solution of the second-order variational differential equation $H(t;s_i,q_i,p)$ that if $\exists j \in \{1,\dots,n_x\}$, $\exists k \in \{1,\dots,(n_x+n_q+n_p)\}$, such that*

$$\begin{aligned}&\underline{H}_{jkk}(t_{i+1};s',q_i,p) \geq 0\\ &\Rightarrow R_k(H_j(t_{i+1};s_{i+1},q_i,p)) - R_k(H_j(t_{i+1};s',q_i,p)) \geq \underline{H}_{jkk}(t_{i+1};s_{i+1},q_i,p) - \underline{H}_{jkk}(t_{i+1};s',q_i,p)\end{aligned} \tag{4.44}$$

and correspondingly if $\exists j \in \{1,\dots,n_x\}$, $\exists k \in \{1,\dots,(n_x+n_q+n_p)\}$, such that

$$\begin{aligned}&\overline{H}_{jkk}(t_{i+1};s',q_i,p) \leq 0\\ &\Rightarrow R_k(H_j(t_{i+1};s_{i+1},q_i,p)) - R_k(H(t_{i+1};s',q_i,p)) \geq \overline{H}_{jkk}(t_{i+1};s_{i+1},q_i,p) - \overline{H}_{jkk}(t_{i+1};s',q_i,p)\,.\end{aligned} \tag{4.45}$$

Using this assumption, we state and prove the following theorem .

Theorem 17 (Adding Multiple Shooting Nodes – Influence on α). *Given Assumption 4 and a problem as defined in Equation* (4.5)*, adding an additional multiple shooting node at $t' \in (t_i,t_{i+1})$ leads to a new convex relaxation. Let the RHS of the second-order variational differential equation be quasi-monotonically increasing in s_i and let Assumption 16 be fulfilled, then*

$$\breve{\alpha}^\dagger \leq \breve{\alpha} \qquad\qquad \breve{\alpha}^\ddagger \leq \breve{\alpha} \tag{4.46a}$$

$$\hat{\alpha}^\dagger \leq \hat{\alpha} \qquad\qquad \hat{\alpha}^\ddagger \leq \hat{\alpha}\,. \tag{4.46b}$$

Proof. The left column of Equation (4.46) follows directly from the fact that the variational differential equations and the initial value $x(t_i) = s_i$ are identical for both, the original problem DMS^N and problem DMS^{N+1} with the additional multiple shooting node at t'. Using the quasi-monotonicity, we derive $\breve{\alpha}^\dagger \leq \breve{\alpha}$ and $\hat{\alpha}^\dagger \leq \hat{\alpha}$ directly from evaluating at $t' < t_{i+1}$.
The idea for proving the other two inequalities is to exploit the constant initial values of sensitivities on each multiple shooting node. By assumption the RHS of the second-order vari-

ational differential equation is quasi-monotonically increasing. Using Theorem 12, we conclude that the trajectories inside of $H(t;s_{i+1},q_i,p)$ do not intersect, in particular this is true for the trajectory bounds with respect to $[\underline{s}_i,\overline{s}_i]$, $[\underline{s}',\overline{s}']$, $[\underline{q}_i,\overline{q}_i]$ and $[\underline{p},\overline{p}]$. Therefore, we have to distinguish between three different cases for each component in the second-order sensitivity tensor individually, depending on whether the constant initial value at t' is contained in the interval $H(t';s_{i+1},q_i,p)$ or not. Figure 4.2 depicts these cases for clarity. Exploiting $H_{jkl}(t';s',q_i,p)=[0,0]$, we obtain that

I if $[0,0]\subseteq H_{jkl}(t';s_i,q_i,p)$, then $H_{jkl}(t;s',q_i,p)\subseteq H_{jkl}(t;s_i,q_i,p)\quad \forall t\in[t',t_{i+1}]$
$\Rightarrow \underline{H}_{jkl}(t_{i+1};s',q_i,p)\geq \underline{H}_{jkl}(t_{i+1};s_i,q_i,p)$ and $\overline{H}_{jkl}(t_{i+1};s',q_i,p)\leq \overline{H}_{jkl}(t_{i+1};s_i,q_i,p)$
$\Rightarrow |H_{jkl}(t_{i+1};s',q_i,p)|\leq |H_{jkl}(t_{i+1};s_i,q_i,p)|$,

II if $[0,0]>\overline{H}_{jkl}(t';s_i,q_i,p)$, then $\underline{H}_{jkl}(t;s',q_i,p)>\overline{H}_{jkl}(t;s_i,q_i,p)\quad \forall t\in[t',t_{i+1}]$
$\Rightarrow \underline{H}_{jkl}(t_{i+1};s',q_i,p)>\underline{H}_{jkl}(t_{i+1};s_i,q_i,p)$ and $\overline{H}_{jkl}(t_{i+1};s',q_i,p)>\overline{H}_{jkl}(t_{i+1};s_i,q_i,p)$
$\Rightarrow |H_{jkl}(t_{i+1};s',q_i,p)|<|H_{jkl}(t_{i+1};s_i,q_i,p)|$,

III if $[0,0]<\underline{H}_{jkl}(t';s_i,q_i,p)$, then $\overline{H}_{jkl}(t;s',q_i,p)<\underline{H}_{jkl}(t;s_i,q_i,p)\quad \forall t\in[t',t_{i+1}]$
$\Rightarrow \underline{H}_{jkl}(t_{i+1};s',q_i,p)<\underline{H}_{jkl}(t_{i+1};s_i,q_i,p)$ and $\overline{H}_{jkl}(t_{i+1};s',q_i,p)<\overline{H}_{jkl}(t_{i+1};s_i,q_i,p)$
$\Rightarrow |H_{jkl}(t_{i+1};s',q_i,p)|<|H_{jkl}(t_{i+1};s_i,q_i,p)|$.

Using the interval version of Gershgorin's circle theorem (Theorem 3) to over- and underestimate α, it is obvious that a smaller absolute value of the off-diagonal entries of the interval Hessians always improves the corresponding α. For negative lower bounds of the diagonal entries this is true as well, but for a positive lower bound on the diagonal and a sufficiently large (especially >0) corresponding radius, we have to ensure that the difference in the radius is larger than the gap between both Gershgorin disc centers. This curvature condition on the second-order sensitivities is expressed in Assumption (4.44) allows us to overestimate the new values $\breve{\alpha}^{\ddagger}_{ijk}$ on $[t',t_{i+1}]$ with the old values $\breve{\alpha}_{ijk}$ at t_{i+1} and we obtain:

$$\begin{aligned}
\breve{\alpha}^{\ddagger}_{ijk} &= \max\left\{0,-\frac{1}{2}\left(\underline{H}_{jkk}(t_{i+1};s',q_i,p)-\sum_{l\neq k}|H_{jkl}(t_{i+1};s',q_i,p)|\right)\right\} && \text{(4.47a)}\\
&\leq \max\left\{0,-\frac{1}{2}\left(\underline{H}_{jkk}(t_{i+1};s_i,q_i,p)-\sum_{l\neq k}|H_{jkl}(t_{i+1};s_i,q_i,p)|\right)\right\} && \text{(4.47b)}\\
&= \breve{\alpha}_{ijk}\quad \forall j,k && \text{(4.47c)}
\end{aligned}$$

and accordingly

$$\begin{aligned}
\hat{\alpha}^{\ddagger}_{ijk} &= \max\left\{0,\frac{1}{2}\left(\overline{H}_{jkk}(t_{i+1};s',q_i,p)+\sum_{l\neq k}|H_{jkl}(t_{i+1};s',q_i,p)|\right)\right\} && \text{(4.48a)}\\
&\leq \max\left\{0,\frac{1}{2}\left(\overline{H}_{jkk}(t_{i+1};s_i,q_i,p)+\sum_{l\neq k}|H_{jkl}(t_{i+1};s_i,q_i,p)|\right)\right\} && \text{(4.48b)}\\
&= \hat{\alpha}_{ijk}\quad \forall j,k\,. && \text{(4.48c)}
\end{aligned}$$

□

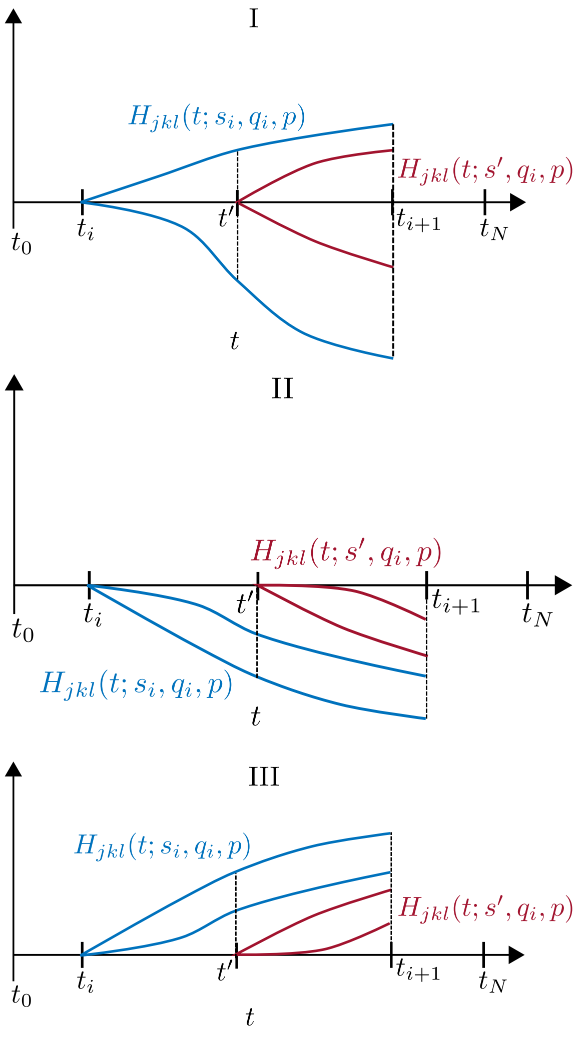


Figure 4.2: An illustration of the three cases in the proof of Theorem 17 indicating the upper and lower bounds on the trajectories of an arbitrarily chosen entry (j,k,l) of the second-order sensitivities $H_{jkl}(t;s',q_i,p)$ and $H_{jkl}(t';s_i,q_i,p)$ between t_i and t_{i+1}.

The constant initial value of the first-order interval sensitivities $G(t;s_i,q_i,p)$ at the multiple shooting nodes will usually further improve the second-order interval sensitivities as those depend on the bounds of the first-order variational differential equations, but it is not necessary to exploit this fact in the proof above. In the numerical results in Chapter 6, we show plots of typical lower and upper bounds for both first- and second-order variational differential equations that underline this improvement due to the constant initial values for practical applications.

Assumption 16 is used to derive the Inequalities (4.47) and (4.48) using the unscaled Gershgorin method. For the scaled Gershgorin approach with a scaling vector d the Assumption can be modified to include d. It remains an open question if this assumption is necessary for other means of under- and overestimating α.

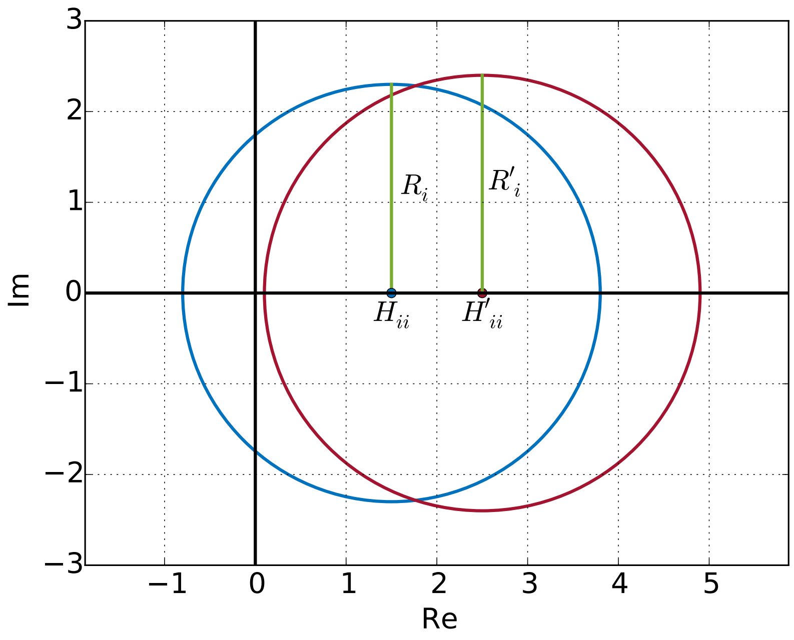


Figure 4.3: An illustration of the case in the proof of Theorem 17, where the underestimation in Equation (3.20) solely based on Gershgorin's theorem is not sufficient.

Theorem 17 is true for exact bounds on the sensitivities. In practical algorithms, we use validated integration as introduced in Section 3.6 which yields an over- respectively underestimation of these bounds. Furthermore, the integrated step-size control that we use may disturb by adding an additional evaluation at t', as it potentially results in different integration grids. To compare the quality, we use the same validated integration method and the same integration steps, in particular, we introduce a forced stop at t'.
This leads to the numerical version of Theorem 17.

Corollary 18 (Adding Multiple Shooting Nodes – Influence of Validated Integrator Stepsize). *Given Assumption 4 and a problem as defined in Equation* (4.5)*, adding an additional multiple shooting node at* $t' \in (t_i, t_{i+1})$ *leads to a new convex relaxation. Let the RHS of the second-order variational differential equation be quasi-monotonically increasing in* s_i*, the bounds on the solution of the differential equation and its first- and second order sensitivities generated by a validated integrator with the same step length in both cases and let Assumption 16 be fulfilled, then*

$$\breve{\alpha}^{\dagger} \leq \breve{\alpha} \qquad \breve{\alpha}^{\ddagger} \leq \breve{\alpha} \tag{4.49a}$$
$$\hat{\alpha}^{\dagger} \leq \hat{\alpha} \qquad \hat{\alpha}^{\ddagger} \leq \hat{\alpha}\,. \tag{4.49b}$$

Proof. For every common validated integrator step at a time t_l $l \in I$ and I the index set of integrator steps between t_i and t_{i+1}, it now still holds true that $|H_{jkl}(t_l;t')| \leq |H_{jkl}(t_l;t_i)|$ $\forall l \in I$ as seen in the proof for Theorem 17. The rest is analogous. □

Regarding the state bounds obtained by the validated integrator and apart from the enforced stop at t' that may vary the taken step size and thus worsen the corresponding bounds, we expect the validated integration to perform better in the multiple shooting case with an increasing amount of multiple shooting nodes. The “reset” in interval width at the multiple shooting nodes as seen in the proof of Theorem 17 helps tremendously to prevent the bounds from blowing up over time which is a typical behavior due to the underlying interval arithmetic. In our numerical results in Chapter 6, we observe this effect multiple times, especially when the validated integration in the single shooting case still fails for certain variable domains, whereas the direct multiple shooting approach already produces valid bounds.

So far, the choice of an additional multiple shooting node $t' \in (t_i, t_{i+1})$ is not restricted. But there is a natural choice, namely, choosing a remaining node of the control grid $\tau' \in (t_i, t_{i+1})$ if such a grid point exists. The next section proves that this choice usually improves the convex relaxation.

4.3.3 Adding Multiple Shooting Nodes on the Control Grid

Adding a multiple shooting node at a time $t' \in [t_i, t_{i+1}]$ that coincides with a remaining control grid point $\tau' \in (t_i, t_{i+1})$ in between t_i and t_{i+1} splits the local discretized control vector $q_i \mathbb{R}^{n_q}$ into two parts $q_i = (q_i^{\dagger^T}, q_i^{\ddagger^T})^T$ with $q_i^\dagger \in \mathbb{R}^{n_{q^\dagger}}$ and $q_i^\ddagger \in \mathbb{R}^{n_{q^\ddagger}}$.

Lemma 19 (Adding Multiple Shooting Nodes on the Control Grid). *Let the assumptions from Theorem 17 hold. Adding the additional multiple shooting node at a control grid point $t' = \tau' \in (t_i, t_{i+1})$, let w.l.o.g. $I^\dagger$ the index set of $q^\dagger$ within $(s_i, q_i, p)^T$ and $I^\ddagger$ the corresponding set of $q^\ddagger$, then $\forall j \in \{1, \dots, n_x\}$ and $\forall (k,l) \in \{(k,l) \in \{1, \dots, n_q\}^2 | k \in I^\dagger \vee l \in I^\dagger\}$*

$$H_{jkl}(t; s_i, q_i, p) = 0 \quad \forall t \in [t_i, t'] \tag{4.50}$$

and

$$H_{jkl}(t; s', q_i, p) = 0 \quad \forall t \in [t', t_{i+1}]. \tag{4.51}$$

Proof. The solution trajectory $x(t; t_i, q_i, p)$ is independent of $q^\dagger \Rightarrow \dot{H}_{jkl}(t; s_i, q_i, p) = 0 \ \forall j \in \{1, \dots, n_x\}$ and $\forall (k,l) \in \{(k,l) \in \{1, \dots, n_q\}^2 | k \in I^\dagger \vee l \in I^\dagger\}$. Together with the initial value $H_{jkl}(t_i; s_i, q_i, p) = 0$, we obtain that

$$H_{jkl}(t; s_i, q_i, p) = 0 \quad \forall t \in [t_i, t'] \tag{4.52}$$

and correspondingly for $q^\ddagger$ on $[t', t_{i+1}]$. $\square$

A direct consequence of this result using Gershgorin’s theorem to under- and overestimate α is that if the corresponding rows were nonzero before the “split” this additional multiple shooting node improves all other Gershgorin disc radii, resulting in a smaller $\|\alpha^\dagger\|$ and $\|\alpha^\ddagger\|$. Additionally, in a practical implementation this multiple shooting grid choice reduces the size of the second-order sensitivity tensor and results in a significant reduction in computation time when determining or updating α.

Therefore, we conclude that with regard to the quality of the convexifications, the natural choice of the multiple shooting grid is the control discretization grid. To elaborate further on this choice, we test different grid combinations in the numerical results in Chapter 6.

Having derived theoretical results on the quality of the direct multiple-shooting-based convexification, in next section we state the full algorithm.

4.4 Direct Multiple-Shooting-Based Global Optimal Control Algorithm

In this section, we use the theoretical results obtained so far to combine them into the full direct multiple-shooting-based αBB algorithm.
We apply the αBB algorithm to obtain a global solution of Problem (2.8). The major difference when applying direct multiple shooting instead of direct single shooting is the introduction of the new variables s_i at the multiple shooting nodes. The matching conditions are treated as described in [PA05] and shown in Equation (2.6) by splitting the equality constraint into two inequality constraints. Please note that this does not significantly increase the computational effort to obtain α, because the corresponding interval Hessian, based on the sensitivities of the ordinary differential equations, have to be computed or overestimated by verified integration on the time horizon in both cases.
In contrast to direct single-shooting-based algorithms that include pointwise constraints on the trajectories, we include the multiple shooting variables as local initial values for the trajectories as described in Section 4.1 and solve the full system simultaneously.
The presented method is complete in the sense that we obtain the global minimum with certainty in infinite run time, but we also know after finite time or with a prescribed tolerance that an approximate global minimizer is found, assuming exact calculations [Neu04]. The proposed implementation is not rigorous, because we do not apply a rigorous NLP solver to solve the arising subproblems, but this part of the algorithm may be replaced by a rigorous solver at the cost of an overall runtime penalty.
The overall framework is a classical spatial branch-and-bound as shown in Algorithm 1 where local solutions to the original problem lead to upper bounds on the global optimum and lower bounds are obtained through the global solutions to the convex relaxations. Furthermore, we branch on the feasible set of the decision variables. And we finally terminate the algorithm when a desired accuracy of the interval on the global objective function value is obtained.
This combination leads to Algorithm 5.
We have obtained theoretical evidence in Sections 4.2 and 4.3 that the convex relaxations based on our novel direct multiple shooting approach are tighter than those based on the classical direct single shooting approach. But when using multiple shooting, we introduce new auxiliary variables s_i at the multiple shooting nodes which may increase the size of our branch-and-bound tree. The next subsection deals with those additional variables.

4.4.1 Treatment of Additional Variables

We have introduced additional variables $s_1, \ldots, s_N$ in comparison to a direct single-shooting-based formulation and using Algorithm 2, we would have to branch on those additional variables as well, leading to a potentially larger branch-and-bound tree. This might remove any computational advantages gained due to the improved relaxations. To remedy this, we introduce a specialized branching strategy in the multiple shooting case. Instead of branching on all decision variables, we branch solely on the original decision variables $q_0, q_1, \ldots, q_{N-1}$ and p, and successively refine the bounds on $s_1, \ldots, s_N$ by propagating the bounds on s_0 through the underlying ODE. First of all, a general bound refinement is essentially for free as the state bounds are integrated once per node using the validated integrator. Even more important is

Algorithm 5: The full multiple-shooting-based αBB algorithm for global optimal control in extension to the single-shooting-based Algorithm 4.

```
Let $\underline{\Phi}$ and $\overline{\Phi}$ denote the bounds on the globally optimal objective value of an
direct-shooting-based discretized OCP (2.8) with $N$ multiple shooting intervals,
$(s,q,p) \in P_0$ the initial domain and $\epsilon$ a desired accuracy for the global optimum.

**Solve** OCP locally on $P_0$ and obtain solution $(s_0^*, q_0^*, p_0^*)$ and $\Phi_\mathrm{M}(x(t_N), p_0)$.
**Simulate** OCP on $P_0$ using a validated integrator along with the first (Equation (2.9))
and second (Equation (2.14)) order variational differential equations to obtain bounds
$S_0$ on the multiple shooting variables and $H_{0,j}$ $j \in \{1,\dots,n\}$ on the second-order
sensitivities for each multiple shooting interval.
**Update** bounds on $s_0$ using $S_0$.
**Relax** OCP on $P_0$ using an $H_{0,j}$ $j \in \{1,\dots,n\}$ based $\alpha_0$ and obtain convex problem
$\mathrm{OCP}_0^\mathrm{cv}$ in the form of Equation (4.5).
**Solve** $\mathrm{OCP}_0^\mathrm{cv}$ locally (=globally) on $P_0$ and obtain solution $(s_0^{\mathrm{cv},*}, q_0^{\mathrm{cv},*}, p_0^{\mathrm{cv},*})$ and
objective value $\Phi_{\mathrm{M},0}(x(t_N), p_0)$.
**Initialize** a set of pairs of domains with corresponding local lower bounds
$\mathcal{P} := \{(P_0, \Phi_{\mathrm{M},0}(x(t_N), p_0))\}$.
**Initialize** bounds $\underline{\Phi} := \Phi_{\mathrm{M},0}(x(t_N), p_0)$, $\overline{\Phi} := \Phi_\mathrm{M}(x(t_N), p_0)$, currently best solution
$(s^*, q^*, p^*) := (s_0^*, q_0^*, p_0^*)$ and an iteration counter $i := 0$.
**while** $\mathcal{P} \neq \emptyset$ **do**
    **if** $\overline{\Phi} - \underline{\Phi} \leq \epsilon$ **then**
        **Stop** algorithm, global solution obtained.
    **end**
    **Increment** iteration counter $i := i + 1$.
    **Select** pair $(P, \Phi_\mathrm{M}(x(t_N), p)) \in \mathcal{P}$.
    **Branch** domain $P$ into partition $Q_1, \dots Q_N$ such that $P = \bigcup_{k \in \{1,\dots,N\}} Q_k$.
    **for** $k = 1$ **to** $N$ **do**
        **Solve** OCP locally on $Q_k$ obtaining $(s_{i,k}^{\mathrm{cv},*}, q_{i,k}^{\mathrm{cv},*}, p_{i,k}^{\mathrm{cv},*})$ and $\Phi_{\mathrm{M},i,k}(x(t_N), p_{i,k})$.
        **if** $\Phi_{M,i,k}(x(t_N), p_{i,k}) < \overline{\Phi}$ **then**
            **Update** upper bound $\overline{\Phi} := \Phi_{\mathrm{M},i,k}(x(t_N), p_{i,k})$ and best known solution
            $(s^*, q^*, p^*) := (s_{i,k}^*, q_{i,k}^*, p_{i,k}^*)$.
        **end**
        **Simulate** OCP on $Q_{i,k}$ with validated integration to obtain $S_{i,k}$ and
        $H_{i,k,j}$ $j \in \{1,\dots,n\}$.
        **Update** bounds on $s_{i,k}$ using $S_{i,k}$.
        **Relax** OCP on $Q_{i,k}$ using an $H_{i,k,j}$ $j \in \{1,\dots,n\}$ based $\alpha_{i,k}$ and obtain convex
        $\mathrm{OCP}_{i,k}^\mathrm{cv}$.
        **Solve** $\mathrm{OCP}_{i,k}^\mathrm{cv}$ locally (=globally) on $Q_{i,k}$ and obtain solution $(s_{i,k}^{\mathrm{cv},*}, q_{i,k}^{\mathrm{cv},*}, p_{i,k}^{\mathrm{cv},*})$
        and objective value $\Phi_{\mathrm{M},i,k}(x(t_N), p_{i,k})$.
    **end**
    **Set** $\mathcal{P} := (\mathcal{P} \setminus (P, \Phi_\mathrm{M}(x(t_N), p))) \bigcup_{k \in \{1,\dots,N\}} (Q_k, \Phi_{\mathrm{M},i,k}(x(t_N), p_{i,k}))$.
    **Set** new global lower bound $\underline{\Phi} := \min\{\Phi_\mathrm{M}(x(t_N), p) : (P, \Phi_\mathrm{M}(x(t_N), p)) \in \mathcal{P}\}$ .
    **Fathom:** remove all pairs with worse local lower bound than the global upper
    bound, i.e. $\mathcal{P} := \mathcal{P} \setminus \{\Phi_\mathrm{M}(x(t_N), p) : (P, \Phi_\mathrm{M}(x(t_N), p)) \in \mathcal{P}, \overline{\Phi} \leq \Phi_\mathrm{M}(x(t_N), p)\}$ .
**end**
The global optimum is now bounded: $\Phi_\mathrm{M}(x(t_N), p_\mathrm{glo}^*) \in [\underline{\Phi}, \overline{\Phi}]$ with $\mathrm{w}([\underline{\Phi}, \overline{\Phi}]) \leq \epsilon$ .
Furthermore, the best solution found $(s^*, q^*, p^*)$ is either the global optimum or has an
objective function value within the desired accuracy.
```

that using Theorem 15, we know that the αBB relaxation of the relaxed matching conditions converge two orders of magnitude faster with respect to $s_1, \dots, s_N$ than the αBB terms in the remaining variables.
We state this strategy as follows.

Lemma 20 (Branching on Free Initial Values, Discretized Controls and Control Values Only). *Given Problem 4.5 and Algorithm 5, it is sufficient to branch only on free initial values s_0, the discretized controls q and the control values p and retain convergence to the global optimum.*

Proof. Branching evenly on s_0, q and p leads to a monotonically decreasing interval width for the bounds on $s_1, \dots, s_N$. Therefore, the assumptions in [PA05] for the convergence of a dynamical αBB-based global optimization remain fulfilled. □

Furthermore, under the assumptions of Theorems 12 and 17, we do not expect the branch-and-bound tree size to be larger, as in every node for a given domain, the direct multiple-shooting-based relaxation is at least as tight. Nevertheless, comparing the branch-and-bound tree sizes directly is difficult, as different domains on the child nodes may lead to a different node selection.
Therefore, using this specialized branching strategy, for practical applications, we expect no additional computational cost with respect to the branch-and-bound tree itself due to these additional variables, but the NLPs at the tree nodes are slightly larger and may be more expensive to solve. Thus, it is important to exploit the highly structured form of those NLPs to minimize the computational overhead [BP84, FSD15].
It remains an open question if we can actually exploit the possibility to branch on those additional variables to speed up the algorithm. Several heuristics in this regard are presented in Chapter 6.

4.4.2 Adaptively Scaled Gershgorin

The following concepts are very general in the sense that they are applicable in non-dynamic global optimization as well. The classical suggestion regarding the choice of a scaling vector d in the scaled Gershgorin method as described in Equations (3.21) and (3.23) is based on the interval width $\mathrm{w}(v) = \overline{v} - \underline{v}$, e.g., [ADFN98]. The author in [Hla15] shows that this choice is optimal in some cases with respect to the maximum separation distance [AMF95] that is defined as the maximum difference between a function $\phi(v) : \mathbb{R}^{n_v} \mapsto \mathbb{R}$ and its convex underestimator $\phi^{\mathrm{cv}}(v) : \mathbb{R}^{n_v} \mapsto \mathbb{R}$ over a domain $[\underline{v}, \overline{v}]$ of v:

$$\max_{v \in [\underline{v}, \overline{v}]} \phi(v) - \phi^{\mathrm{cv}}(v) \,. \tag{4.53}$$

For the αBB convex underestimator we obtain

$$\max_{v \in [\underline{v}, \overline{v}]} \phi(v) - \phi^{\mathrm{cv}}(v) = \max_{v \in [\underline{v}, \overline{v}]} \sum_{i=0}^{n_v - 1} -\alpha_i (\overline{v}_i - v_i)(\underline{v}_i - v_i) = \sum_{i=0}^{n_v - 1} \alpha_i \frac{(\overline{v}_i - \underline{v}_i)^2}{4} \,. \tag{4.54}$$

For the case that $\mathrm{w}(v)$ is not optimal, the author in [Hla15] proposes two iterative algorithms that optimize the scaling vector d with respect to the maximum separation distance of the convex relaxation. We refer to the second one as “iteratively scaled Gershgorin method” and have reimplemented the approach for a direct numerical comparison in Section 6.5 in the context of global optimal control and especially in comparison with the scaled Gershgorin method.

The proposed algorithms are iterative methods and have to be applied separately for the over- and underestimation of the eigenvalues for each relaxed matching condition that are necessary for the direct multiple-shooting-based algorithm. Therefore, we propose to heuristically perform only a single iteration of the first algorithm with a specific index selection. To give reasons for the heuristic and the specific selection, we present a different and more specifically a direct proof of the property that each iteration preserves the sparsity pattern of α. Furthermore, we interpret the results visually and measure the performance of our heuristic, compared to the scaled Gershgorin method with the variable interval width as scaling and both iterative methods in the context of global optimal control in Chapter 6. To differentiate from the full iterative methods, we refer to our proposal as "adaptively scaled Gershgorin method".

We start with the observation that it is highly desirable to detect convexity and concavity if it is possible at all using the scaled Gershgorin method. A function is convex if $\alpha^{\text{cv}} = 0$ and concave if $\alpha^{\text{cc}} = 0$. Therefore, instead of the maximum separation distance that takes into account not only α, but also the complete under- or overestimation term, we propose to focus on the sparsity pattern of α.

Definition 21 (Sparsity pattern). *The sparsity pattern of a vector $\alpha \in \mathbb{R}^n_\alpha$ is the index set I such that $\alpha_i = 0$.*

Let there be a Gershgorin disc $D(\underline{H}_{ii}, R_i(H))$ that lies already inside the right half-plane, i.e., $\underline{H}_{ii} - R_i(H) > 0$. We would not have to push this diagonal entry during our relaxation. On the other hand, we can use the scaling to increase the radius R_i using the scaling vector d such that the disc touches the imaginary axis exactly once in 0. This leads to an optimal scaling in the sense that it improves the determined α without losing the sparsity pattern of the unscaled Gershgorin method. Figure 4.4 visualizes the idea behind this adaptively scaled Gershgorin method.

We state and prove this result formally as

Lemma 22 (Adaptively Scaled Gershgorin Method). *Let an interval matrix $H \in [\mathbb{R}]^{n\times n}$ and corresponding $\alpha^{\text{cv}} \in \mathbb{R}^n$ based on the unscaled interval Gershgorin method be given. If $\exists j \in 1,\dots,n$, such that $\underline{H}_{jj} - R_j(H) > 0$, then defining $i := \arg\min_j \{\frac{R_j(H)}{\underline{H}_{jj}} \mid \underline{H}_{jj} - R_j(H) > 0\}$ and scaling the interval Gershgorin method with*

$$d_i := \frac{1}{\underline{H}_{ii}} \qquad\qquad d_j := \frac{1}{R_i(H)} = \frac{1}{\sum_{k\neq i} |H_{ik}|} \quad \forall j \neq i \tag{4.55}$$

results in the new α'^{cv} being at least as good: $\alpha'^{\text{cv}}_k \leq \alpha^{\text{cv}}_k \quad \forall k = 0,\dots,n$.

Proof. In contrast to the indirect proof given in [Hla15], we prove this lemma directly and obtain for the component i

$$\underline{H}_{ii} - \sum_{j\neq i} |H_{ij}| \frac{d_j}{d_i} = \underline{H}_{ii} - \sum_{j\neq i} |H_{ij}| \frac{\underline{H}_{ii}}{\sum_{j\neq i} |H_{ij}|} = 0 \quad . \tag{4.56}$$

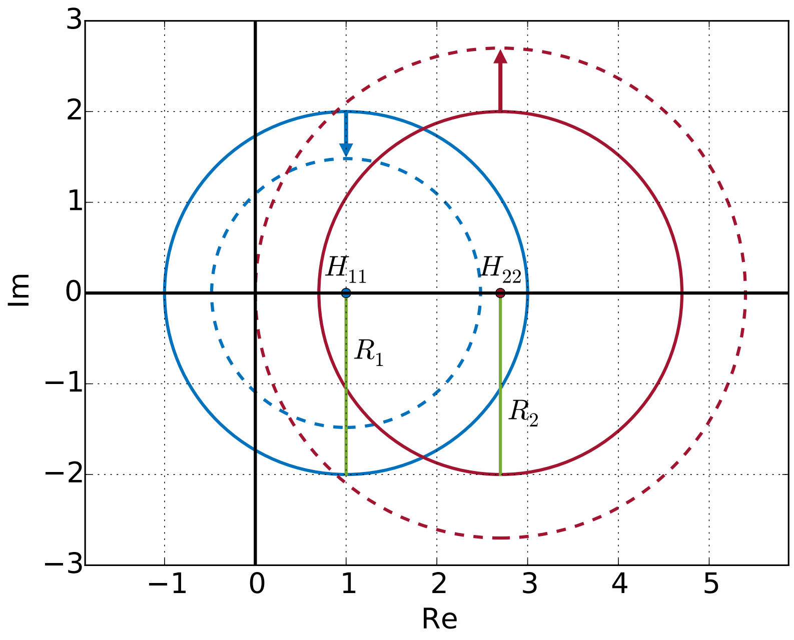


Figure 4.4: An illustration of the idea of Lemma 22. The scaling is improved such that the smallest positive Gershgorin disc is increased such that it touches zero and the radius of every other Gershgorin disc is reduced by that fraction.

Therefore, we maintain $\alpha'^{\text{cv}} = \alpha^{\text{cv}} = 0$, but can not scale this particular component any further without loosing the sparsity pattern of α^{cv}. By assumption, we have $\underline{H}_{ii} > 0$ and we obtain

$$\underline{H}_{ii} - R_i(H) > 0 \Rightarrow 1 > \frac{R_i(H)}{\underline{H}_{ii}} > 0\,. \tag{4.57}$$

Using this, for all $j \neq i$ it holds true that

$$\begin{aligned}
\underline{H}_{jj} - \sum_{k\neq j} |H_{jk}| \frac{d_k}{d_j} &= \underline{H}_{jj} - \sum_{\substack{k\neq j\\k\neq i}} \left(|H_{jk}| \frac{R_i(H)}{R_i(H)} \right) - |H_{ji}| \frac{R_i(H)}{\underline{H}_{ii}} && \text{(4.58a)}\\
&= \underline{H}_{jj} - \sum_{\substack{k\neq j\\k\neq i}} \left(|H_{jk}| \right) - |H_{ji}| \frac{R_i(H)}{\underline{H}_{ii}} && \text{(4.58b)}\\
&\geq \underline{H}_{jj} - \sum_{\substack{k\neq j\\k\neq i}} |H_{kj}| - |H_{ji}| && \text{(4.58c)}\\
&= \underline{H}_{jj} - \sum_{k\neq j} |H_{kj}| \quad . && \text{(4.58d)}
\end{aligned}$$

And thus $\alpha'^{\text{cv}}_k \leq \alpha^{\text{cv}}_k \quad \forall i = 0, \ldots, n$.
The specific choice $i := \arg\min_j \{ \frac{R_j(H)}{\underline{H}_{jj}} \mid \underline{H}_{jj} - R_j(H) > 0 \}$ is not a necessary assumption, but is our heuristic that aims to maximize the gap in the underestimation in Equation (4.58c) above and is a valid choice due to $\underline{H}_{ii} > 0$. □

This result is independent of the optimal control algorithm and can be used in the single shooting approach and in the context of global optimization of NLPs as well.
The next step is to analyze when equality occurs and when true inequality holds true. In our version of the proof, we see that we obtain $\|\alpha'^{\text{cv}}\| = \|\alpha^{\text{cv}}\|$ if and only if $|H_{ji}| = 0 \quad \forall j \neq i$. In other words, the adaptively scaled Gershgorin method leads to tighter relaxations as soon as a single off-diagonal element, that is not in the row respectively column of the selected scaling component i, is not zero. Therefore, we expect the adaptively scaled Gershgorin method to outperform the unscaled version almost every time. We formalize this result as

Corollary 23 (Adaptively Scaled Gershgorin Method – True Inequality). *Let an interval matrix $H \in [\mathbb{R}]^{n\times n}$ and corresponding $\alpha^{\text{cv}} \in \mathbb{R}^n$ based on the unscaled interval Gershgorin method be given. If $\exists j \in 1,\dots,n$, such that $\underline{H}_{jj} - R_j(H) > 0$, then defining $i := \arg\min_j\{\frac{R_j(H)}{\underline{H}_{jj}} \mid \underline{H}_{jj} - R_j(H) > 0\}$ and scaling the interval Gershgorin method with*

$$d_i := \frac{1}{\underline{H}_{ii}} \qquad\qquad d_j := \frac{1}{R_i(H)} = \frac{1}{\sum_{k\neq i}|H_{ik}|} \quad \forall j \neq i \tag{4.59}$$

results in the new α'^{cv} being better: $\alpha'^{\text{cv}}_k < \alpha^{\text{cv}}_k \quad \forall k = 0,\dots,n$, if and only if $\exists j \neq i$, such that $|H_{ij}| \neq 0$.

Proof. In Equation (4.58a), we can use the additional assumption $\exists j \neq i, \text{s.t.} |H_{ij}| \neq 0$. We obtain

$$\begin{aligned}
\underline{H}_{jj} - \sum_{k\neq j} |H_{jk}| \frac{d_k}{d_j} &= \underline{H}_{jj} - \sum_{\substack{k\neq j\\k\neq i}} \big(|H_{jk}|\big) - |H_{ji}| \frac{R_i(H)}{\underline{H}_{ii}} && (4.60a)\\
&> \underline{H}_{jj} - \sum_{\substack{k\neq j\\k\neq i}} |H_{kj}| - |H_{ji}| && (4.60b)\\
&= \underline{H}_{jj} - \sum_{k\neq j} |H_{kj}| \quad . && (4.60c)
\end{aligned}$$

And thus $\alpha'^{\text{cv}}_k < \alpha^{\text{cv}}_k \quad \forall k = 0,\dots,n$.
In the other direction, if $|H_{ij}| = 0, \forall j \in \{1,\dots,n\}$ then the inequality above becomes an equality and we obtain $\alpha'^{\text{cv}}_k = \alpha^{\text{cv}}_k \quad \forall k = 0,\dots,n$. □

Furthermore and in particular for our direct multiple-shooting-based global optimization algorithm, this adaptively scaled Gershgorin method offers an adapted scaling for the under- and overestimation of the eigenvalue independently. We formalize this as

Corollary 24 (Adaptively Scaled Gershgorin Method – Concave Case). *Let an interval matrix $H \in [\mathbb{R}]^{n\times n}$ and corresponding $\alpha^{\text{cc}} \in \mathbb{R}^n$ based on the unscaled interval Gershgorin method be given. If $\exists j \in 1,\dots,n$, such that $\overline{H}_{jj} + R_j(H) < 0$, then defining $i := \arg\max_j\{\frac{R_j(H)}{\overline{H}_{jj}} \mid \overline{H}_{jj} + R_j(H) < 0\}$ and scaling the interval Gershgorin method with*

$$d_i := \frac{1}{\overline{H}_{ii}} \qquad\qquad d_j := \frac{1}{R_i(H)} = \frac{1}{\sum_{k\neq i}|H_{ik}|} \quad \forall j \neq i \tag{4.61}$$

results in the new α'^{cc} being better: $\alpha'^{\text{cc}}_k < \alpha^{\text{cc}}_k \quad \forall k = 0,\dots,n$, if and only if $\exists j \neq i$, such that $|H_{ij}| \neq 0$.

Proof. $\overline{H}_i i$ is negative by assumption and we obtain $-1 < \frac{R_j(H)}{\overline{H}_{jj}} < 0$, resulting in our heuristic to be $i := \arg\max_j\{\frac{R_j(H)}{\overline{H}_{jj}} \mid \overline{H}_{jj} + R_j(H) < 0\}$. The rest is analogous to the proof of Lemma 22 and Corollary 23. □

So far, for the sake of simplicity, we preserved the sparsity pattern of the unscaled Gershgorin method. We can extend the results by combining our adaptively scaled Gershgorin method with other choices of the scaling vector d to improve them as well to optimality with respect to the respective sparsity pattern. The obvious choice for such a scaling combination is the interval width $\mathrm{w}(v)$ that is already optimal with respect to the maximum separation distance in some cases. To simplify the notation, we define the radius that is already scaled by d as $R^d_j(H) := \sum_{j\neq i} |H_{ij}| \frac{d_j}{d_i}$ and state

Corollary 25 (Combining the Adaptively Scaled Gershgorin Method with Other Scalings). *Let an interval matrix $H \in [\mathbb{R}]^{n\times n}$ and corresponding $\alpha^{\text{cv}} \in \mathbb{R}^n$ based on the scaled interval Gershgorin method with a scaling vector d be given. If $\exists j \in 1,\dots,n$, such that $\underline{H}_{jj} - R_j(H) > 0$, then defining $i := \arg\min_j\{\frac{R_j(H)}{\underline{H}_{jj}} \mid \underline{H}_{jj} - R_j(H) > 0\}$ and scaling the interval Gershgorin method with*

$$d_i := \frac{1}{\underline{H}_{ii}} \qquad\qquad d_j := \frac{1}{R_i(H)} = \frac{1}{\sum_{k\neq i}|H_{ik}|} \quad \forall j \neq i \tag{4.62}$$

results in the new α'^{cv} being better: $\alpha'^{\text{cv}}_k < \alpha^{\text{cv}}_k \quad \forall k = 0,\dots,n$, if and only if $\exists j \neq i$, such that $|H_{ij}| \neq 0$.

Proof. Analogous to the proof of Lemma 22 and Corollary 23. □

Finally, we can combine the results of Corollary 24 with other scalings accordingly. This is the version of the scaled Gershgorin version that we use to determine α in the numerical results for both the single shooting and multiple shooting approach. Especially in our multiple shooting-based approach this gives us the means for fast and independently improved under- and overestimations that lead to potentially tighter bounds. We present a direct numerical comparison between the scaled, the full iteratively scaled and the our adaptively Gershgorin methods in Section 6.5.
We close this section with the observation that improving the sparsity pattern of α by scaling a Gershgorin disc that overlaps with the left half-plane, but whose center is in the right half-plane, is often not possible. We would have to shrink the corresponding radius and in the process increase the radius of the other discs, resulting in a potentially worse α. As Figure 4.5 shows α may even loose the sparsity pattern in other components.

4.4.3 Reduced Space Relaxations

For an at least partially fixed initial value, we can use the fixed values to reduce the variable space for the two point direct multiple shooting approach by performing the convex relaxation only in the directions of the discretized controls and control values. Without loss of generality

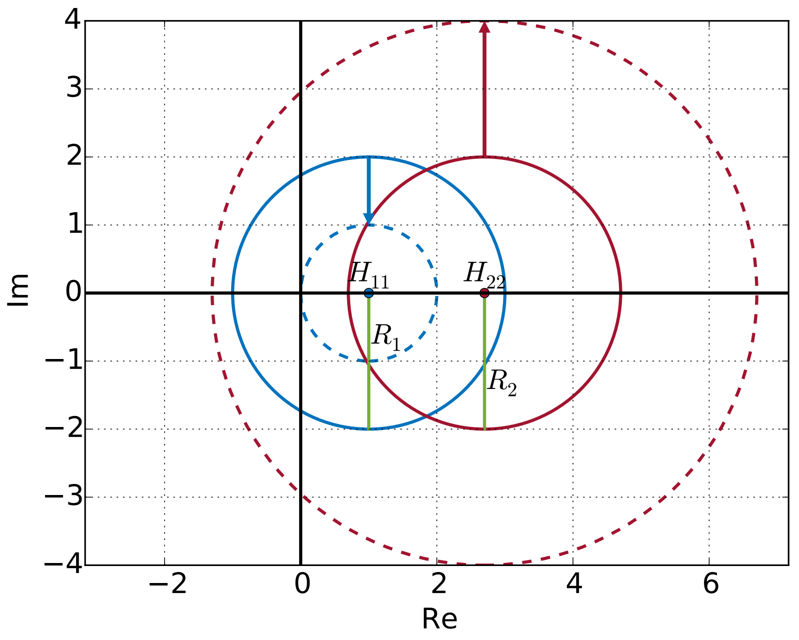


Figure 4.5: An illustration why decreasing the radius of a Gershgorin disc, such that one additional component of α becomes zero may change the sparsity pattern.

and to simplify the notation, we assume all initial values s_0 to be fixed. Although using the αBB relaxation, those fixed values have no direct influence on the convex relaxation as the corresponding quadratic term becomes zero, they have an indirect influence as they increase the size of the second-order sensitivities. Using the scaled Gershgorin approach to underestimate the eigenvalues, this leads to bigger Gershgorin disc radii for the other components and thus potentially worse α^q and α^p.

For the two point multiple shooting approach, this results in a similar formulation as performed in [EF00b] and [PA02] for the inclusion of constraints or additional objective function evaluations at intermediate points $t_j \in [t_0, t_N]$.

In this section, we want to go further and propose a heuristic for our direct multiple-shooting-based global optimal control algorithm for more multiple shooting nodes. This heuristic is motivated by the results of Theorem 15 that the αBB relaxation with respect to s_1 to s_N converges two orders faster in the interval widths of s_0, q and p than the remaining quadratic terms. Assuming fixed initial values s_0, this is fostered by the observation that the convergence rate improves further, because $[\underline{s}_0, \bar{s}_0] = 0$ and the corresponding terms and mixed terms in the convergence rate vanish. Futhermore, this coincides with the experience that in practical applications the direct multiple shooting approach is said to be more robust against being stuck in local minimas. Although it is always possible to create counterexamples, we show this effect once more on the numerical test cases from the literature in Chapter 6. Figure 4.6 illustrates the reduced space approach.

We formalize the reduced space direct multiple-shooting-based convexification of the original problem in Equation 2.8 and in particular in comparison to the full space approach described in Equation 4.5:

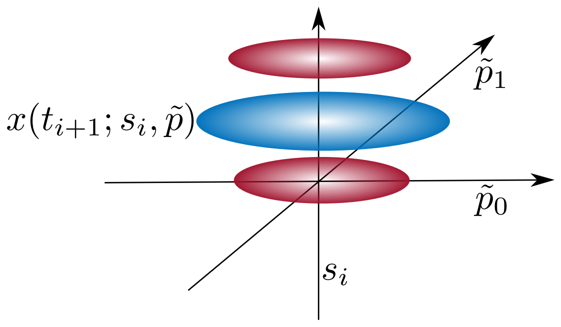


Figure 4.6: An illustration of the space reduction described in Section 4.4.3. We still convexify the matching conditions with respect to the discretized control variables q and the control values p, here combined into a parameter vector $\tilde{p} := (q, p)$, but not with respect to the additional multiple shooting variables s_i in the lifted space. The gradient of each disc represents the convexity of a one dimensional state at t_{i+1} with respect to $\tilde{p}$ for a fixed s_i.

$$\begin{aligned}
\min_{s,q,p} \quad & \Phi_{\mathrm{M}}(s_N, p) + \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \Phi_{\mathrm{L}}(t, x(t; t_i, s_i, q_i, p), \tilde{u}_i(t, q_i), p)\, \mathrm{d}t && \text{(4.63a)}\\
\text{s.t.} \quad & 0 \geq x_j(t_{i+1}; t_i, s_i, q_i, p) - s_{i+1,j} && \text{(4.63b)}\\
& \quad + \sum_{k=1}^{n_q} \check{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k} - q_{i,k})(\underline{q}_{i,k} - q_{i,k}) + \sum_{k=1}^{n_p} \check{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k - p_k)(\underline{p}_k - p_k) && \text{(4.63c)}\\
& 0 \geq -x_j(t_{i+1}; t_i, s_i, q_i, p) + s_{i+1,j} && \text{(4.63d)}\\
& \quad + \sum_{k=1}^{n_q} \hat{\alpha}^{\mathrm{q}}_{ijk}(\overline{q}_{i,k} - q_k)(\underline{q}_{i,k} - q_{i,k}) + \sum_{k=1}^{n_p} \hat{\alpha}^{\mathrm{p}}_{ijk}(\overline{p}_k - p_k)(\underline{p}_k - p_k) && \text{(4.63e)}\\
& 0 \leq c^{\mathrm{cc}}(t_i, s_i, \tilde{u}_i(t_i, q_i), p) && \text{(4.63f)}\\
& 0 \leq r^{\mathrm{eq,cc}}(s_0, \ldots, s_N, p) && \text{(4.63g)}\\
& 0 \leq -r^{\mathrm{eq,cv}}(s_0, \ldots, s_N, p) && \text{(4.63h)}\\
& 0 \leq r^{\mathrm{ieq,cc}}(s_0, \ldots, s_N, p) && \text{(4.63i)}\\
& \quad \forall i \in \{0, \ldots, N-1\}, j \in \{0, \ldots, n_x\}\,. && \text{(4.63j)}
\end{aligned}$$

This is a heuristic for $N > 2$ and therefore, we use this reduced space approach only in the two-point direct multiple shooting or when directly stated otherwise in Chapter 6.

For only partially fixed initial values, let n_p' the number of free initial values. We parametrize these free initial values, using an additional parameter vector p' and define the augmented parameter vector, including the original control values and the additional free initial values as $\tilde{p} := \begin{pmatrix} p \\ p' \end{pmatrix}$ with dimension $n_{\tilde{p}}$. The solution trajectory of the ODE $F(q, \tilde{p}) : \mathbb{R}^{n_q, n_{\tilde{p}}} \to \mathbb{R}^n_x$ is then a function of q and $\tilde{p}$ only. For the details on such ODE reformulations, we refer to detailed descriptions in textbooks such as [Har02].

4.4.4 Fast Bounds on the Second-Order Sensitivities

One computationally expensive aspect of the αBB approach when applied to optimal control problems is the necessity of bounds on the second-order sensitivities. One straight forward method is to integrate the first- and second-order variational differential equations (2.9) and (2.14) together with the differential equation itself.

To give an impression of the computational effort, using our novel direct multiple-shooting-based global optimization algorithm with a piecewise constant control discretization per multiple shooting node and assuming that both, the control and the multiple shooting grid, are the same, e.g. $\tau_0 = t_0, \tau_1 = t_1, \ldots \tau_N = t_N$ in Equations (2.2) and (2.5), we obtain that based on the state dimension n_x, the number of control functions n_u and control parameters n_p, we derive the number of first-order variational differential equations to be $n_{\text{fos}} = n_x(n_x+n_u+n_p)$ and the dimension of the second-order sensitivities as $n_{\text{sos}} = \frac{1}{2}n_x(n_x + n_u + n_p)(n_x + n_u + n_p + 1)$ as the latter are symmetric in each state dimension. The total number of trajectories to be integrated to calculate the interval sensitivities is therefore the sum $n_{\text{valint}} = n_x + n_{\text{fos}} + n_{\text{sos}}$.

A first straight forward approach to speed up the overall computational effort is based on the observation that, based on Equation (3.4), it is obvious that the convex relaxations become tighter by branching on the variables alone, resulting in improved bounds and therefore smaller $(\overline{x}_i - x_i)(\underline{x}_i - x_i)$ with a constant α. Reusing any valid α from the parent node, it is possible to update α only after a finite amount of branches without interfering with the convergence theory. Depending on the problem, the optimal number of updates may vary and for the larger applications in Chapter 6, we test several different parameter sets regarding this delayed α update in the context of global optimal control.

Another point of attack to improve the computational effort of the second-order sensitivities is the validated integrator. We are able to speed up the calculation of the corresponding bounds significantly if we exploit the usage of Taylor-expansion-based validated integrators. In [Bar06], the authors calculate first-order sensitivities using a Taylor-based integration. We are interested in the bounds on first- and second-order sensitivities instead. Furthermore, higher order enclosure methods [NJP01] such as the one implemented in `VSPODE` rely on the first-order sensitivities already. Applying them to the system of second-order variational differential equations effectively results in third-order sensitivities being calculated where synergies could be used. Although the application is straightforward using automatic differentiation (AD) tools, the computational effort is significant. Therefore, we propose to integrate the generation of second-order sensitivity bounds directly into the validated integrator, using the Taylor model to not only provide validated bounds on the states, but also on the second-order sensitivities.

Given the current integration time $t_j \in [t_i, t_{i+1}]$ together with a step size h_j and using already the final enclosure of k-th order, based on a-priori bounds $X(t; t_j, q_i, p)$ on the solution $x(t)$, we obtain bounds on the solution trajectory using the high-order Taylor model as seen in Equation (3.27b) as follows:

$$\begin{aligned} x(t;s_i,q_i,p) &\subseteq x(t_j) + \sum_{l=1}^{k-1}(t-t_j)^l f^{[l]}(x(t_j)) + h_j^k f^{[k]}(X(t_j)) \\ &\quad + \left(I + \sum_{l=1}^{k-1} h_j^l \frac{\partial f^{[l]}}{\partial x}(X(t_j)) \right)(X(t_j) - x(t_j)) , \qquad (4.64) \\ x(t) &\in X(t) \quad \forall t \in [t_j, t_{j+1}] . \qquad (4.65) \end{aligned}$$

To ease the notation, we combine the local control discretization q_i and control values p into a local parameter vector $\tilde{p}_i := (q_i, p)^T$. We calculate the first and second-order derivatives of the Taylor coefficients with respect to s_i and obtain for the first integration step at the corresponding multiple shooting interval, e.g. $t_j = t_i$ as

$$\begin{aligned} \frac{\mathrm{d}x(t;s_i,q_i,p)}{\mathrm{d}s_i} &\subseteq I + \sum_{l=1}^{k-1}(t-t_j)^l \frac{\mathrm{d}f^{[l]}(x(t_j))}{\mathrm{d}s_i} + h_j^k \frac{\mathrm{d}f^{[k]}(X(t_j))}{\mathrm{d}s_i} \\ &\quad + \frac{\mathrm{d}}{\mathrm{d}s_i}\left(\left(I + \sum_{l=1}^{k-1} h_j^l \frac{\partial f^{[l]}}{\partial x}(X(t_j)) \right)(X(t_j) - x(t_j)) \right) , \qquad (4.66) \\ x(t) &\in X(t) \quad \forall t \in [t_j, t_{j+1}] \qquad (4.67) \end{aligned}$$

and for $t_j \in (t_i, t_{i+1}]$

$$\begin{aligned} \frac{\mathrm{d}x(t;s_i,q_i,p)}{\mathrm{d}s_i} &\subseteq \sum_{l=1}^{k-1}(t-t_j)^l \frac{\mathrm{d}f^{[l]}(x(t_j))}{\mathrm{d}s_i} + h_j^k \frac{\mathrm{d}f^{[k]}(X(t_j))}{\mathrm{d}s_i} \\ &\quad + \frac{\mathrm{d}}{\mathrm{d}s_i}\left(\sum_{l=1}^{k-1} h_j^l \frac{\partial f^{[l]}}{\partial x}(X(t_j)) \right)(X(t_j) - x(t_j)) , \qquad (4.68) \\ x(t) &\in X(t) \quad \forall t \in [t_j, t_{j+1}] . \qquad (4.69) \end{aligned}$$

This distinction between the multiple shooting node itself at t_i and the rest of interval is not necessary for the first-order sensitivities with respect to $\tilde{p}$ and we derive

$$\begin{aligned} \frac{\mathrm{d}x(t;s_i,q_i,p)}{\mathrm{d}\tilde{p}} &\subseteq \sum_{l=1}^{k-1}(t-t_j)^l \frac{\mathrm{d}f^{[l]}(x(t_j))}{\mathrm{d}\tilde{p}} + h_j^k \frac{\mathrm{d}f^{[k]}(X(t_j))}{\mathrm{d}\tilde{p}} \\ &\quad + \frac{\mathrm{d}}{\mathrm{d}\tilde{p}}\left(\sum_{l=1}^{k-1} h_j^l \frac{\partial f^{[l]}}{\partial x}(X(t_j)) \right)(X(t_j) - x(t_j)) , \qquad (4.70) \\ x(t) &\in X(t) \quad \forall t \in [t_j, t_{j+1}] . \qquad (4.71) \end{aligned}$$

Correspondingly the second-order sensitivities are

$$\frac{d^2x(t;s_i,q_i,p)}{d(s_i,\tilde{p})^2} \subseteq \sum_{l=1}^{k-1}(t-t_j)^l \frac{d^2 f^{[l]}(x(t_j))}{d(s_i,\tilde{p})^2} + h_j^k \frac{d^2 f^{[k]}(X(t_j))}{d^2(s_i,\tilde{p})}$$

$$+ \frac{d^2}{d(s_i,\tilde{p})^2}\left(\sum_{l=1}^{k-1} h_j^l \frac{\partial f^{[l]}}{\partial x}(X(t_j))\right)(X(t_j)-x(t_j)) \tag{4.72}$$

$$x(t) \in X(t) \quad \forall t \in [t_j, t_{j+1}]\,. \tag{4.73}$$

Using AD, the necessary derivatives of the right-hand side function f can be evaluated very efficiently and accurately .
This method speeds up the single shooting algorithm as well as our multiple-shooting-based approach and therefore it is not directly relevant for a comparison. Most results in Chapter 6 are obtained by integrating the full variational differential equations. The reason for this choice is that to take advantage of these fast and efficient bounds on the second-order sensitivities, the method should be integrated directly into the validated integrator. This allows to prevent unnecessary higher-order derivatives by reusing them for different purposes and to take the width of the remainder term into account during the step size control. Finally, most validated integrators employ means to counteract the so-called wrapping effect. This is important for the interval bounds of the sensitivities as well in a practical implementation. The numerical results in Section 6.6 are obtained by a prototype implementation of the method above to give a first impression of the efficiency, but we do not expect to reach the full potential with respect to accuracy and computational time, compared to an implementation integrated into the validated integrator itself.

Chapter 5

The Software Package GloOptCon

In extension to the theoretical results from the last chapter, all proposed algorithms and methods are implemented in the novel software package GloOptCon in order to provide numerical results that validate the theoretical predictions from Chapter 4 for a number of test examples from the literature, as well as for real-world problems. Furthermore, Algorithm 4 and 3 are implemented in GloOptCon as well. This allows a direct and close comparison with the direct single shooting approach and is necessary as currently, there is no software package freely available that solves OCPs using direct single-shooting-based on an αBB convexification. Furthermore, other global optimal control packages such as [Sin04] are not maintained anymore making a comparison difficult without a reimplementation of the global optimization algorithms.
The goal of GloOptCon is to provide for the first time one efficient platform for global optimal control algorithms that employ external state-of-the-art nonlinear program (NLP) solvers, integrators, validated integrators, automatic differentiation (AD) tools and interval arithmetic libraries in a flexible way. As part of the algorithms, the software package includes fully featured efficient direct single and multiple shooting solvers for the local optimization of optimal control problems (OCPs), which additionally supports parallelization. Furthermore, a number of analysis and plotting capabilities ease the interpretation and visualization of the results.
Whereas this chapter focuses solely on the software package GloOptCon itself, Chapter 6 shows the numerical results.

5.1 Goals and Structure

Our goal is to develop a novel software package that is able to efficiently solve OCPs of the form presented in Equation (2.1) globally using direct single as well as multiple-shooting-based on Algorithms 3, 4 and 5, which also features parallelization. This common code base allows us to make a direct comparison of the computation time in absence of significant differences resulting from more or less optimized implementations.
Furthermore, and especially due to the size of such a software project, it is important to reuse as many state-of-the-art components developed and provided from other workgroups as possible. A common interface allows the usage of different NLP solvers, integrators, validated integrators, AD tools and interval libraries. This allows to focus on implementing the multiple-shooting-based algorithm and admits an interface verification by applying different external libraries for a specific test problem. To allow a wide range of external libraries and a fast code core, the programming language C++ is chosen. As it simplifies some code, the current version uses the C++11 standard of the language.

Last but not least, in the light of computationally expensive global optimal control and the trend to more and more CPU cores, we want to show the advantages of multiple shooting when parallelizing the method.
To summarize the demands on our software package, we require

- an efficient direct single and direct multiple-shooting based αBB implementations to solve OCPs,
- interfaces to allow usage of standard components for NLP solvers, integrators, validated integrators and AD tools,
- numerical verifications of the theoretical results from Chapter 4 and
- a parallelization for the direct multiple shooting version.

The software package has a class structure centered around a central class that reserves the necessary memory, contains the implemented algorithms and interfaces to the user supplied model and any external libraries necessary. Switching between single shooting, a different number of multiple shooting nodes, another control discretization, an alternative integrator or solvers and many more features is done by single calls of this central class.
Problems are set up through source files as shown in Figure 5.1. These user-generated source files must supply template-based versions of their objective and right-hand side functions. Those files set up the problem, reserve the necessary memory by suppling the dimensions and set the function pointers afterwards. The next step is to supply bounds or fixed values for the problem. Calling the member function `solve()` locally optimizes the problem, whereas calling `solve_global()` starts the global optimization. A number of additional member functions ease to choose a different solver, integrator or validated integrator or modify the method of choice for obtaining the derivatives necessary during the optimization. Please note that it is also possible to omit supplying differential equations and instead provide an objective function and general nonlinear constraints depending on a control value vector p only, resulting in the αBB-based global optimization of the NLP provided.
To visualize the results, GloOptCon features a number of plot capabilities ranging from a Gnuplot interface to visualize the solution trajectories as shown in Figure 5.2 to the possibility to write the branch-and-bound structure into a GraphViz file that generates visualizations as shown in Figure 5.3. More member functions to generate and control different plots are briefly described in Figure 5.4.

5.2 Algorithms, Interfaces and Dependencies

One core principle of the development of GloOptCon is to use state-of-the art libraries for for central necessary features. Therefore, the software package relies on a number of external libraries. Despite those dependencies compilation flags allow a fast deactivation of components that are not necessary or available. It is possible to rely on external open source software only by using `Ipopt` as the required NLP solver, the validated integrator from `ACADO`, `CppAD` for automatic differentiation and the supplied Runge-Kutta method for integration. The following sections give a brief overview of the core algorithms, as well as of the implemented interface alternatives for every external component.

```
#include "dms.hpp"

const double a[5]  = { 8.86, 24.25, 23.67, 18.75, 20.70 };
const double bR[5] = { 10215.4, 18820.5, 17008.9, 14190.8, 15599.8 };

template<typename T>
T mfcn(const T *t, const T *x, const T *p) {
    return -x[1];
}

template<typename T>
void ffcn(const T *t, const T *x, const T *u, const T *p, T *xdot) {
    T k[5];
    for ( int i = 0; i < 5; ++i ) {
        k[i] = exp( a[i] - bR[i] * u[0] / 698.15 );
    }

    xdot[0] = - k[0] * x[0]                - ( k[2] + k[3] + k[4] ) * x[0] * x[1];
    xdot[1] =   k[0] * x[0] - k[1] * x[1]                + k[2]   * x[0] * x[1];
}

int main() {
    int NX = 2;
    int NU = 1;
    int NP = 0;
    int mshoot = 5;

    dms global4(NX, NU, NP, mshoot, 0, 1, 0, 1);
    SET_FCN( global4, mfcn, ffcn, nullptr )

    global4.set_t(10.0);
    global4.fix('x', 0, 0, 1.0);
    global4.fix('x', 1, 0, 0.0);
    global4.set_a('u', 0,  698.15/748.15, 698.15/748.15, 1.0);

//  global4.solve();
    global4.solve_global();

    global4.plot_gnuplot();

    return 0;
}
```

Figure 5.1: Minimal code example for the oil pyrolysis application described in Equation (6.7).

Global Optimization Framework

GloOptCon contains a deterministic global optimization framework including a spatial branch-and-bound based on Algorithm 1. On each node the corresponding lower and upper bounds of the variables and the α vector is saved. Optionally the solution, the relaxed solution and the corresponding dual variables can be stored for warmstart purposes. Furthermore, there are several branching heuristics and strategies available, especially the specialized treatment of the additional multiple shooting variables as described in Section 4.4.1. As a heuristic for tests of deterministic global optimization algorithms an implementation of a stochastic multistart approach is available as well.

Direct Multiple-Shooting-Based αBB

The core of GloOptCon is the efficient implementation of the direct multiple-shooting-based global optimization approach described in Algorithm 5. It is possible to include the matching

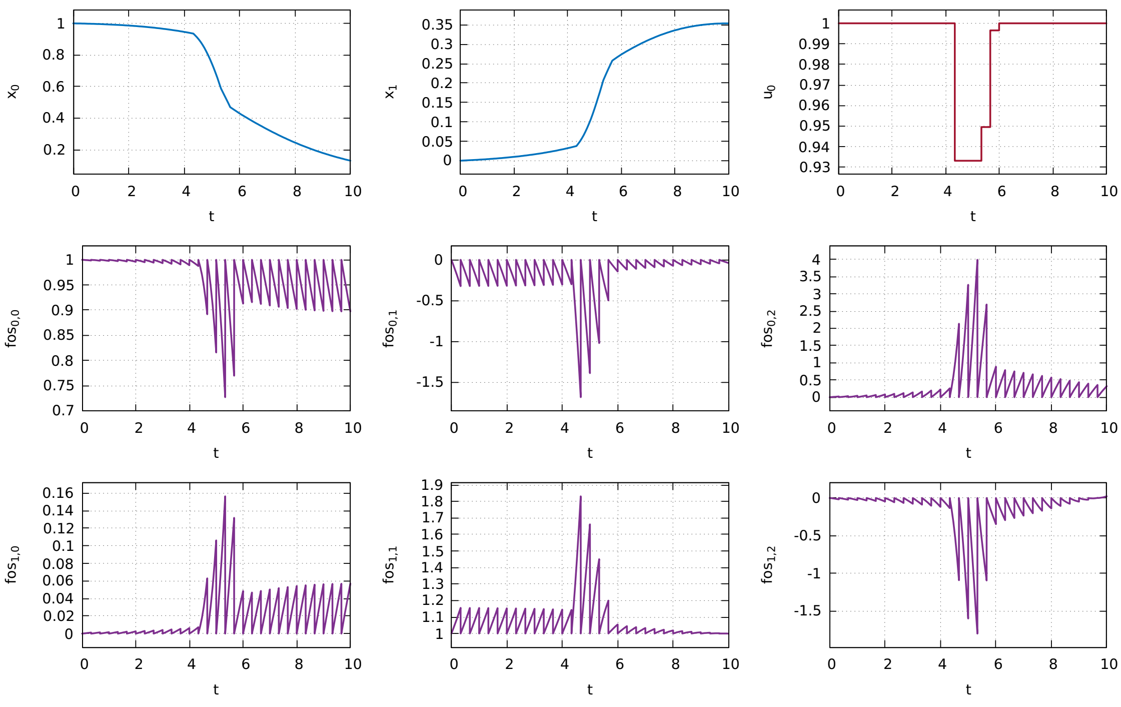


Figure 5.2: The software package has a Gnuplot interface to visualize the solutions. In this figure a local solution to Problem (6.7) is shown with trajectories x_0 and x_1 and a control function u_0 that is discretized locally constant on 30 intervals. Finally, the trajectories of the local first-order sensitivities of the direct multiple shooting approach are labeled with fos. Please note that the independent sensitivities over successive multiple shooting intervals are combined into single jagged trajectories to reduce the number of plots necessary.

conditions either as equality constraints for the non-relaxed problem or as inequality constraints with the αBB relaxations as seen in Equation 4.5. The same works for the treatment of general nonlinear constraints and nonlinear Mayer-type objective functions. For all necessary derivatives a fallback and crosscheck exists using finite differences. Furthermore, the typical block structure of the derivatives if fully exposed to the NLP solver interface to make it possible to exploit this pattern efficiently.

Direct Single Shooting Alternatives

With a single flag, it is possible to deactivate the multiple shooting discretization to obtain the direct single-shooting-based optimal control algorithm for a comparison. Although there are some significantly varying code parts, especially regarding the relaxation of a Mayer-type objective function, the sensitivities and the missing matching conditions due to the differences in the algorithms, all common parts share the same code, allowing a comparison with as little influence from code optimizations as possible.

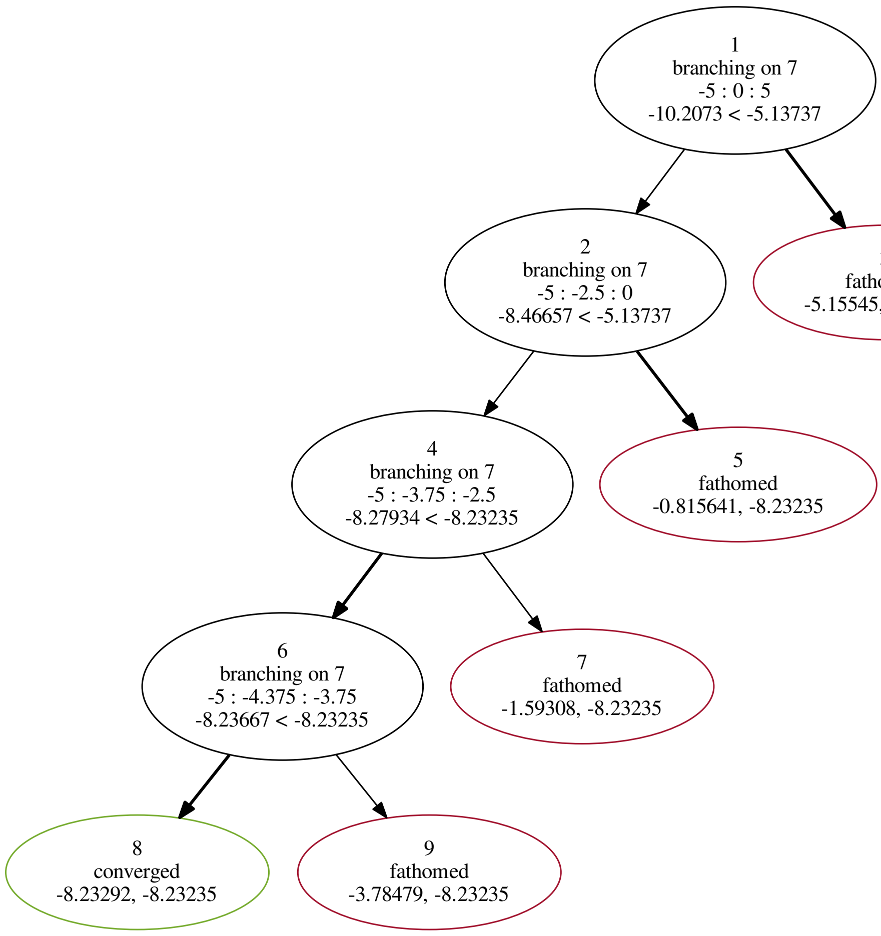


Figure 5.3: The software package GloOptCon is able to generate source files for `GraphViz` [EGK$^+$02] to visualize the Branch-and-Bound tree.

```
void plot_gnuplot(const std::string &filename = "");
void plot_write(const std::string &filename = "") const;

void plot_add(unsigned int index, const std::string &name = "");
void plot_add_xup();
void plot_add_fos();
void plot_add_sos();
void plot_remove(unsigned int index);

void plot_obj(int index_a, int index_b = -1, const std::string &name = "");
void plot_bnb();
```

Figure 5.4: Calling `plot_gnuplot` directly results in a standard plot including the state trajectories and control functions over time. This can be adjusted to include any trajectory, including first and second-order sensitivities. Furthermore, it is possible to visualize the branch-and-bound in up to two dimensions and the objective function with respect to one or two variables.

Adaptively Scaled Gershgorin Method

Our novel adaptively scaled Gershgorin technique presented in Section 4.4.2 is the method of choice for the under- and overestimation of the eigenvalues of the second-order interval sensitivities. To give a numerical performance comparison in this case as well, the scaled Gershgorin method with different scaling choices and the first algorithm presented in [Hla15] are provided by GloOptCon as well.

Nonlinear Program Solvers

`Ipopt` [WB06] as an interior point (IP) solver and `filterSQP` [FL98, FL02] from the class of sequential quadratic programming (SQP) solvers provide means for solving the resulting NLP after discretization in an efficient way. Both libraries allow the possibility to exploit the highly structured problem from Equation (2.8) using a sparsity pattern. Despite the fact that we are not able to apply the condensing technique [BP84] in the relaxed case, because the matching constraints are split into inequality constraints in the relaxed problem as seen in Equation (4.5), this allows a fast solution of both, the original as well as the relaxed problem formulation. Figure 5.5 shows an unified modeling language (UML) diagram of the solver interface.

Integrators

The integrator interface currently supplies a fast, self-written, error controlled, explicit Runge-Kutta method for non-stiff problems, a Fortran routine from MUSCOD-II implementing an explicit Runge-Kutta method [Lei99] and `CVODES` from the `SUNDIALS` suite [HBG+05b] for stiff differential equations providing Adams-Moulton and backward-differentiation formula (BDF) implementations. All interfaces are capable of not only providing the state trajectories, but also the corresponding first and second-order sensitivities based on different ways of generating the derivatives. Furthermore, a dummy integrator is supplied to provide means to disable time-dependent states and controls. This allows the user to supply only a Mayer-type objective function and general nonlinear constraint based on control values p resulting in an non-dynamic NLP that can be solved using αBB in combination with the different Gershgorins methods including our novel suggestions from Section 4.4.2. Figure 5.6 shows an UML diagram of the integrator interface of GloOptCon.

Validated Integrators

For the crucial validated integration of the states and the second-order sensitivities GloOptCon provides interfaces to `VSPODE` [LS07b], `VNODE` [NJP01] and the method presented in [HVC13] that is implemented in the `ACADO` toolkit [HFD11a]. All validated algorithms have different advantages and disadvantages. As mentioned in Section 3.6, `VSPODE` usually provides the tightest bounds at the cost of computational time, especically for a larger control discretization. `VNODE` provides very good bounds as well and scales better for larger problems. The bounds provided by the `ACADO` toolkit implementation where not as tight for our applications, but reasonably fast and the implementation is available as open-source. As a crosscheck and fallback method GloOptCon provides a sampling-based integrator that uses one of the standard integration methods to generate the necessary bounds on the states and second-order

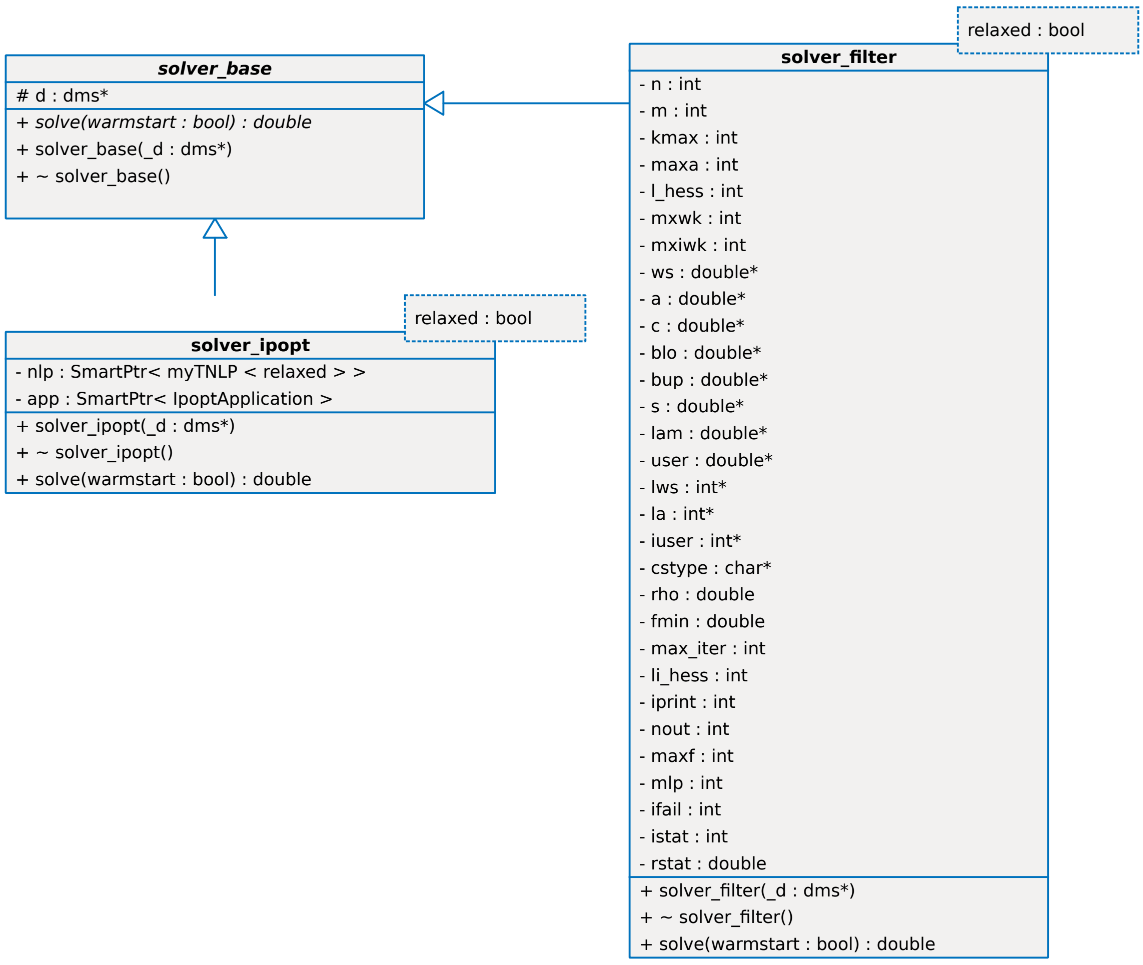


Figure 5.5: UML model of the NLP solver interface. The interface is implemented as a template deciding that decides if it is a solver for the original or the relaxed problem that has a different dimension and structure.

sensitivities. Please note that this approach is not validated and a heuristic only as the resulting bounds may not include all possible values. Figure 5.7 shows an UML diagram of the validated integrator interface.

Automatic Differentiation

The derivatives necessary for normal and interval-valued functions can be obtained using the automatic differentiation tools `CppAD` [Bel15] and `FADBAD++` [BS96]. The latter one is used in `VSPODE` and `VNODE` as well to generate the necessary Taylor expansions. Both are available as open-source and provide means for efficient and exact computation of the derivatives used in the various global optimization algorithms, integrators and NLP solvers. As a fallback and crosscheck method, finite difference approximations are available for all function evaluations as well.

Interval Arithmetic

Whereas the GloOptCon software package itself tries to limit the reliance on external interval arithmetic tools, `VNODE` uses either `PROFIL/BIAS` [Knü94] or `filib++` [LTW+06] for the underlying interval arithmetic. Both are able to take into account potential rounding errors, provide a large number of basic mathematical functions and are easy to combine with the AD tools introduced above. In the presence of a nonlinear Mayer-type objective function or general nonlinear constraints, we use `filib++` in combination with `CppAD` or `FADBAD++` to evaluate the corresponding interval Hessians.

5.3 Verification

Significant care is utilized in GloOptCon to verify the results. As seen in the last section, most interfaces have at least two different options to crosscheck the external results and interface implementations themselves. Furthermore, the GloOptCon includes a number of tests to verify the function evaluations and their derivatives. The test that can be applied to any implemented model include

- `test_int()` to test the state integration with all linked integrators and their interfaces,
- `test_fos()` to test the first-order sensitivities with all linked integrators and their interfaces,
- `test_sos()` to test the second-order sensitivities with all linked integrators and their interfaces,
- `test_con()` to test the constraint evaluation and corresponding derivatives
- `test_intstates()` to compare the state bounds obtained through the linked validated integrators with the result from the random sampling approach,
- `test_intsos()` to compare the interval sensitivity obtained through the linked validated integrators with the result from the random sampling approach and
- `test_conv()` to test all relaxations for convexity by calculating the eigenvalues of the second derivative on a large number of random sampling points.

The sampling tests for the validated integrators utilize the custom `valint_multi` integrator. With $N^{\text{multistart}}$ being the number of sampling points, $H_i^{\text{multistart}}$ the corresponding second-order interval sensitivity and $\alpha_i^{\text{multistart}}$ the corresponding α obtained, and we have to check that

$$H_i \supseteq H_i^{\text{multistart}} \quad \forall i \in \{1, \dots N^{\text{multistart}}\} \quad \text{and} \tag{5.1}$$

$$\alpha_i \leq \alpha_i^{\text{multistart}} \quad \forall i \in \{1, \dots N^{\text{multistart}}\} \quad . \tag{5.2}$$

A similar heuristic test is available for the global optimization algorithms itself. The results of a multistart approach where a local optimization is started at random points over the domain of s, q and p can be compared to the bounds on the true global optimum obtained with our algorithms.

5.4 Parallelization

The direct multiple shooting approach decouples the integration over several smaller time intervals. Therefore, a parallel integration is possible on top of any other potential parallelization, especially in the branch-and-bound [GC94]. The parallelization in GloOptCon is done using `OpenMP` [Ope13] and the results in Table 5.1 and Figure 5.8 show the efficiency of our parallel direct multiple shooting approach used in the local optimization of Problem (6.5) with 30 locally constant control discretizations.

Let t_i^{r} be the runtime of the solution using i parallel cores, then we define the strong scaling efficiency [Kam15] as the runtime speedup for the same problem divided by the number of cores utilized, that is

$$\frac{t_1^{\mathrm{r}}}{i t_i^{\mathrm{r}}}\,. \tag{5.3}$$

number of cores	mean value of runtime over 10 runs	strong scaling efficiency
1	0.0972505	1.0
2	0.0523415	0.9289999331
3	0.0374260	0.8661581076
4	0.0316758	0.7675457289
5	0.0261216	0.7445983401
6	0.0238592	0.6793361331

Table 5.1: Efficiency of the parallelized direct multiple shooting approach tested on Problem (6.4) with 30 locally constant control discretizations. The first column shows the number of cores used, whereas the second row is the mean value of the computational time over 10 runs. The last row shows the strong scaling efficiency as defined in Equation (5.3).

On top of the parallelization of the integration and with the price of more memory usage, it is possible to parallelize the branch-and-bound as well. Please note that the maximum number of straight forward usable threads is the product of both, the number of multiple shooting intervals and the branch-and-bound threads, because the multiple shooting parallelization takes place at each node individually.

Using our novel direct multiple shooting algorithm, we are able to parallelize the validated integration of the second-order sensitivities as well. This can be done in two ways: a one-pass method, where we reuse the bounds on s from the parent node and directly start the parallel integration on each multiple shooting node individually, or a two-pass method, where we integrate the states only at first in a serial fashion and then use these bounds in a second pass to start the parallel second-order sensitivity integrations. As the integration of the full second-order variational differential equations takes significant more time then the integration of the states only, the second method still retains a high parallel efficiency.

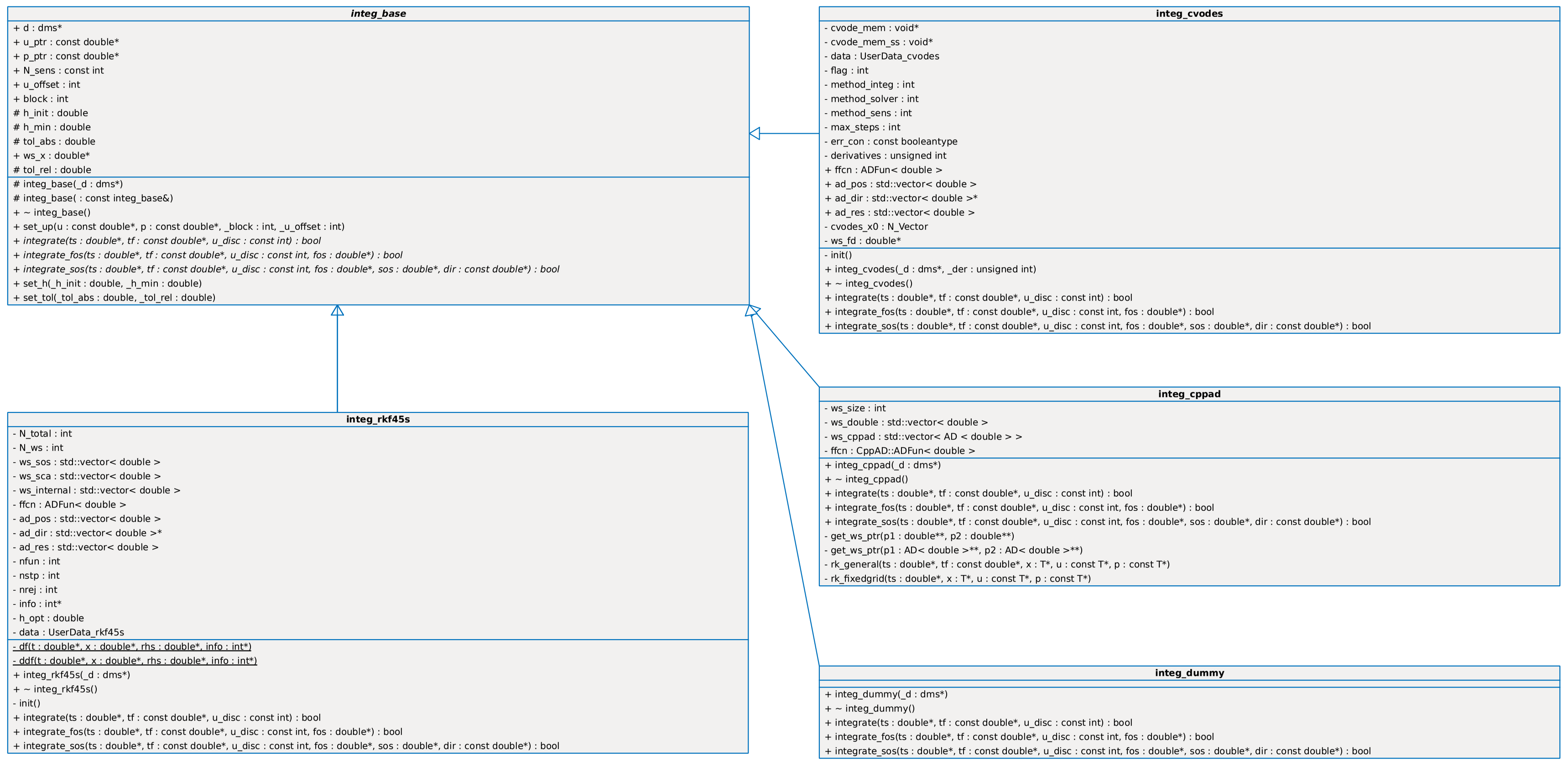

integ_base
+ d : dms*
+ u_ptr : const double*
+ p_ptr : const double*
+ N_sens : const int
+ u_offset : int
+ block : int
# h_init : double
# h_min : double
# tol_abs : double
+ ws_x : double*
# tol_rel : double
# integ_base(_d : dms*)
# integ_base(: const integ_base&)
+ ~ integ_base()
+ set_up(u : const double*, p : const double*, _block : int, _u_offset : int)
+ integrate(ts : double*, tf : const double*, u_disc : const int) : bool
+ integrate_fos(ts : double*, tf : const double*, u_disc : const int, fos : double*) : bool
+ integrate_sos(ts : double*, tf : const double*, u_disc : const int, fos : double*, sos : double*, dir : const double*) : bool
+ set_h(_h_init : double, _h_min : double)
+ set_tol(_tol_abs : double, _tol_rel : double)
integ_cvodes
- cvode_mem : void*
- cvode_mem_ss : void*
- data : UserData_cvodes
- flag : int
- method_integ : int
- method_solver : int
- method_sens : int
- max_steps : int
- err_con : const booleantype
- derivatives : unsigned int
+ ffcn : ADFun< double >
+ ad_pos : std::vector< double >
+ ad_dir : std::vector< double >*
+ ad_res : std::vector< double >
- cvodes_x0 : N_Vector
- ws_fd : double*
- init()
+ integ_cvodes(_d : dms*, _der : unsigned int)
+ ~ integ_cvodes()
+ integrate(ts : double*, tf : const double*, u_disc : const int) : bool
+ integrate_fos(ts : double*, tf : const double*, u_disc : const int, fos : double*) : bool
+ integrate_sos(ts : double*, tf : const double*, u_disc : const int, fos : double*, sos : double*, dir : const double*) : bool
integ_cppad
- ws_size : int
- ws_double : std::vector< double >
- ws_cppad : std::vector< AD < double > >
- ffcn : CppAD::ADFun< double >
+ integ_cppad(_d : dms*)
+ ~ integ_cppad()
+ integrate(ts : double*, tf : const double*, u_disc : const int) : bool
+ integrate_fos(ts : double*, tf : const double*, u_disc : const int, fos : double*) : bool
+ integrate_sos(ts : double*, tf : const double*, u_disc : const int, fos : double*, sos : double*, dir : const double*) : bool
- get_ws_ptr(p1 : double**, p2 : double**)
- get_ws_ptr(p1 : AD< double >**, p2 : AD< double >**)
- rk_general(ts : double*, tf : const double*, x : T*, u : const T*, p : const T*)
- rk_fixedgrid(ts : double*, x : T*, u : const T*, p : const T*)
integ_rkf45s
- N_total : int
- N_ws : int
- ws_sos : std::vector< double >
- ws_sca : std::vector< double >
- ws_internal : std::vector< double >
- ffcn : ADFun< double >
- ad_pos : std::vector< double >
- ad_dir : std::vector< double >*
- ad_res : std::vector< double >
- nfun : int
- nstp : int
- nrej : int
- info : int*
- h_opt : double
- data : UserData_rkf45s
- df(t : double*, x : double*, rhs : double*, info : int*)
- ddf(t : double*, x : double*, rhs : double*, info : int*)
+ integ_rkf45s(_d : dms*)
+ ~ integ_rkf45s()
- init()
+ integrate(ts : double*, tf : const double*, u_disc : const int) : bool
+ integrate_fos(ts : double*, tf : const double*, u_disc : const int, fos : double*) : bool
+ integrate_sos(ts : double*, tf : const double*, u_disc : const int, fos : double*, sos : double*, dir : const double*) : bool
integ_dummy
+ integ_dummy(_d : dms*)
+ ~ integ_dummy()
+ integrate(ts : double*, tf : const double*, u_disc : const int) : bool
+ integrate_fos(ts : double*, tf : const double*, u_disc : const int, fos : double*) : bool
+ integrate_sos(ts : double*, tf : const double*, u_disc : const int, fos : double*, sos : double*, dir : const double*) : bool

Figure 5.6: UML model of the integrator interface. Please note that the member variable encapsulation is not very strict due to limitations in the interface to external Fortran code.

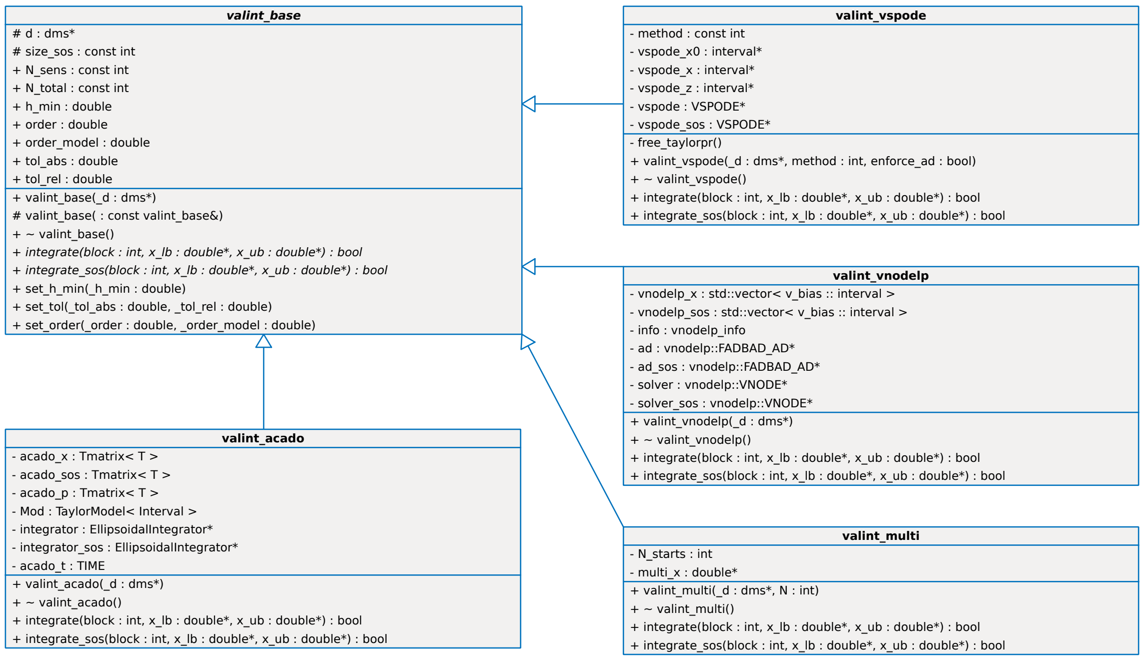

valint_base
# d : dms*
# size_sos : const int
+ N_sens : const int
+ N_total : const int
+ h_min : double
+ order : double
+ order_model : double
+ tol_abs : double
+ tol_rel : double
+ valint_base(_d : dms*)
# valint_base(: const valint_base&)
+ ~ valint_base()
+ integrate(block : int, x_lb : double*, x_ub : double*) : bool
+ integrate_sos(block : int, x_lb : double*, x_ub : double*) : bool
+ set_h_min(_h_min : double)
+ set_tol(_tol_abs : double, _tol_rel : double)
+ set_order(_order : double, _order_model : double)
valint_vspode
- method : const int
- vspode_x0 : interval*
- vspode_x : interval*
- vspode_z : interval*
- vspode : VSPODE*
- vspode_sos : VSPODE*
- free_taylorpr()
+ valint_vspode(_d : dms*, method : int, enforce_ad : bool)
+ ~ valint_vspode()
+ integrate(block : int, x_lb : double*, x_ub : double*) : bool
+ integrate_sos(block : int, x_lb : double*, x_ub : double*) : bool
valint_vnodelp
- vnodelp_x : std::vector< v_bias :: interval >
- vnodelp_sos : std::vector< v_bias :: interval >
- info : vnodelp_info
- ad : vnodelp::FADBAD_AD*
- ad_sos : vnodelp::FADBAD_AD*
- solver : vnodelp::VNODE*
- solver_sos : vnodelp::VNODE*
+ valint_vnodelp(_d : dms*)
+ ~ valint_vnodelp()
+ integrate(block : int, x_lb : double*, x_ub : double*) : bool
+ integrate_sos(block : int, x_lb : double*, x_ub : double*) : bool
valint_acado
- acado_x : Tmatrix< T >
- acado_sos : Tmatrix< T >
- acado_p : Tmatrix< T >
- Mod : TaylorModel< Interval >
- integrator : EllipsoidalIntegrator*
- integrator_sos : EllipsoidalIntegrator*
- acado_t : TIME
+ valint_acado(_d : dms*)
+ ~ valint_acado()
+ integrate(block : int, x_lb : double*, x_ub : double*) : bool
+ integrate_sos(block : int, x_lb : double*, x_ub : double*) : bool
valint_multi
- N_starts : int
- multi_x : double*
+ valint_multi(_d : dms*, N : int)
+ ~ valint_multi()
+ integrate(block : int, x_lb : double*, x_ub : double*) : bool
+ integrate_sos(block : int, x_lb : double*, x_ub : double*) : bool

Figure 5.7: UML model of the validated integrator interface.

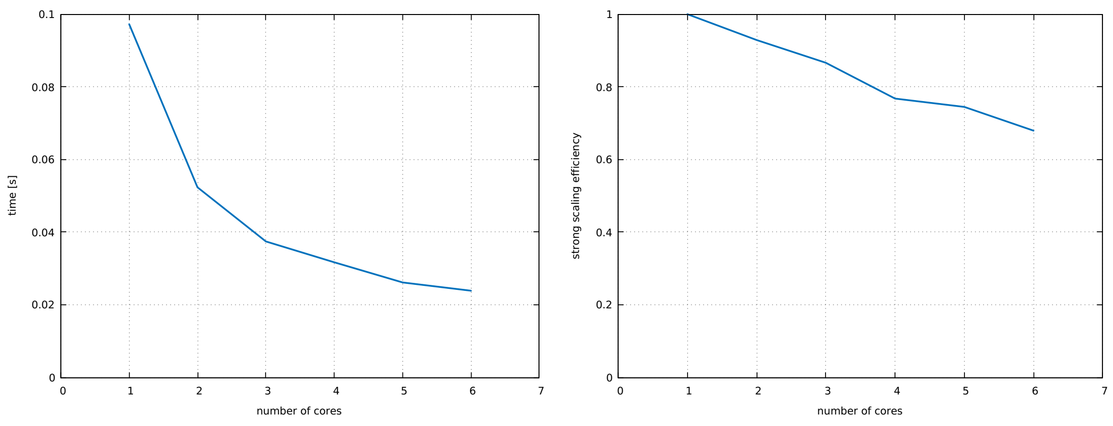


Figure 5.8: Plots of the results from Table 5.1. The left plot shows the decreasing computational time with increasing number of cores and the right plot shows the strong scaling efficiency of the direct multiple shooting approach with parallelized integration. The slight bumb in efficiency for four cores is due to the fact that the 30 multiple shooting nodes used in this test can not be distributed equally onto the four cores.

Part III

Numerical Results

Chapter 6

Numerical Results

In this chapter, we use our software package GloOptCon from Chapter 5 to apply the algorithm derived in Chapter 4 to various test problems from the global optimal control literature and applications, where multiple local minima were observed.

We begin with two problems containing control values only. The analytical solution for the first problem is known and therefore, we can verify our solutions with the exact one. Even in the absence of discretized controls, we expect our direct multiple shooting algorithm to be an improvement over the direct single shooting approach taking into account our theoretical results in Section 4.3. Furthermore, the problems give an important indication on the computational overhead for solving the underlying nonlinear problems (NLPs) using direct multiple shooting.

Afterwards, we apply our algorithms to two common test problems from the literature that contain control functions and show that our theoretical results directly translate to a better performance in practice. The theory from Section 4.3.3 predicts an increasing efficiency for finer common control and multiple shooting grids. As those test problems are rather small, we continue with the global optimization of a robot application where multiple local minima were observed and which is novel in the context of global optimal control. Furthermore, it is the first optimal control problem (OCP) in our tests with an underlying boundary value problem (BVP), in contrast to the other problems from the literature containing no constraints on the states at the final time.

The last application in this chapter is a mixed integer optimal control problem. This problem class is significantly more complicated and additionally the problem considered contains boundary constraints at the end time as well, posing a significant challenge.

In the next sections, we take a closer look at our adaptively scaled Gershgorin method introduced in Section 4.4.2. We furthermore quantify the potential gain of the fast bounds on the second-order sensitivities derived in Section 4.4.4 using a special wrapper around a validated integrator. We show that the bounds obtained using this method are significantly faster.

To sum up, the theoretically proven increase of efficiency is validated by the numerical results in this chapter. Compared to the previous approach from the literature, the number of iterations for typical problems is more than halved, meanwhile the computation time is reduced by almost an order of magnitude. This in turn allows the global solution of larger optimal control problems.

6.1 Numerical Comparison

As presented in Chapter 5, the software implementation GloOptCon is a custom direct single and multiple shooting code written in C++. To obtain the best possible comparison, no other optimizations such as custom treatment of linear or bilinear terms are utilized. Furthermore, no special branching rules or heuristics were used, apart from the special branching for multiple shooting as derived in Section 4.4.1.

We start by giving a short overview of the test problems and applications. Due to the first four problems being common in the global optimization literature, we only state the formulation and give references for further information on the particular problem. The robot application, which is novel in the context of global optimal control and larger than the other problems, and the aircraft model that is a mixed integer optimal control problem with boundary constraints are described in more detail.

The numerical results in this chapter were obtained using our software implementation with its custom direct single and multiple shooting code written in C++ as presented in Chapter 5. If not stated otherwise, we apply the adaptively scaled Gershgorin method from Section 4.4.2 to the second-order interval sensitivities obtained using `VSPODE` 1.4 [LS07b]. In our direct multiple shooting algorithm, we branch on the control parametrization vector and the control values only as described in Section 4.4.1. Regarding the choice of the branching index, we use the largest interval widths. The regular integration and sensitivity generation is performed using our custom AD-based explicit Runge-Kutta (RK) method for non-stiff problems and with the backward differentiation formula (BDF) implementation from `CVODES` from the `SUNDIALS` software suite in version 2.6.0 [HBG$^+$05a] for all stiff differential equations. The underlying nonlinear problems (NLP) are solved using `Ipopt` 3.12.3 [WB06] for the global optimal control algorithms and `filterSQP` during the multistart heuristics to identify different local minima. The latter proved to be more robust with respect to bad initial values that occur using random values within a large domain for the multiple shooting variables, discretized controls and control values for the applications in Sections 6.3 and 6.4. The automatic differentiation (AD) is performed using cppad [Bel15] and all computational times are measured using an Intel Core i7-5820K at 4Ghz with 16GB of memory.

6.2 Test Problems From The Literature

We begin the presentation of our numerical results using several small test problems from the global dynamic optimization literature. They range from containing only a single state and one control value without a continuous control to larger ones based on several control variables.

The problem dimensions are based on the number of states n_x, the number of control discretization parameters per multiple shooting node n_q and the number of control values n_p. The number of independent variables n_{var} is different for single shooting and multiple shooting. Any free initial value is contained in $n_{\tilde{p}}$ in addition to the control values in the single shooting case, whereas they are automatically contained in n_x in the multiple shooting case. Finally, n_{fos} is the size of the first-order sensitivities and n_{sos} the dimension added by the second-order of sensitivities. Please note that due to the symmetry of the second-order sensitivities, the amount is reduced to the entries of the lower triangular matrices. The total number of trajectories to be integrated to calculate the interval sensitivities is therefore the sum n_{int}.

$$n_{\text{var}} = \begin{cases} n_u + n_{p'} & \text{using single shooting} \\ n_x + n_u + n_p & \text{using multiple shooting} \end{cases} \tag{6.1a}$$

$$n_{\text{fos}} = n_x n_{\text{sens}} \tag{6.1b}$$

$$n_{\text{sos}} = n_x n_{\text{sens}}(n_{\text{sens}} + 1)/2 \tag{6.1c}$$

$$n_{\text{int}} = n_x + n_{\text{fos}} + n_{\text{sos}} \,. \tag{6.1d}$$

Table 6.1 gives an overview of the problem dimensions using multiple shooting with a piece-wise constant control discretization per multiple shooting interval. Furthermore, we assume that the control and the multiple shooting grid are identical, i.e. $\tau_0 = t_0, \tau_1 = t_1, \dots \tau_N = t_N$ in Equations (2.2) and (2.5).

	n_x	n_u	n_p	n_{fos}	n_{sos}	n_{int}
Illustrative Example	1	0	1	2	3	6
Negative Resistance Circuit	2	0	2	8	20	30
Singular Control Problem	4	1	0	20	60	84
Oil Shale Pyrolysis	2	1	0	6	12	20
Robot Application	4	2	0	24	84	112
Aircraft Application	3	1	0	12	30	45

Table 6.1: First and second-order sensitivity system sizes for validated integration using our direct multiple-shooting-based global optimal control algorithm with locally constant control discretization q and equivalent control discretization and multiple shooting grids.

6.2.1 Problems with Control Values

Illustrative Example

We start with a test problem from the literature containing only a single state x and a single control value p where an analytical solution is still available. The following problem is considered among others in [CL04] and in [LS07a] as “Illustrative Example” and goes back to [PA02]. It is the smallest example we solve. It does not contain any time dependent controls, but only one time independent control value p and therefore, it does not suffer from any discretization error with respect to the control parametrization used in direct optimal control methods. Even though it would be trivial to obtain exact bounds on the states, no manual optimizations were performed to measure the performance of the algorithm as objective as possible. The algorithms are applied without exploiting problem specific simplifications or optimizations often used in global optimization.

The problem formulation is the following:

$$\min_{p} \quad -x(t_N)^2 \tag{6.2a}$$

$$\text{s.t.} \quad \dot{x}(t) = -x(t)^2 + p \tag{6.2b}$$

$$x(t_0) = 9 \tag{6.2c}$$

$$t \in [t_0, t_N] = [0, 1] \tag{6.2d}$$

$$p \in [-5, 5]\,. \tag{6.2e}$$

The problem is concave on $p \in [-5, 5]$ and therefore has two local minima $\Phi(-5) \approx -5.39433$ and $\Phi(5) \approx -8.23262$ at the boundaries. Figure 6.1 shows an approximation of the state bounds on the time interval from $[t_0, t_N] = [0, 1]$ as well as a visualization of the objective function with respect to p. The global solution obtained using Algorithm 5 and an $\epsilon = 10^{-3}$ coincides with the lower bound of all trajectories over the domain of p in the left plot of Figure 6.1 and trajectory based on the worse local solution at $p = 5$ is equivalent to the upper bound.

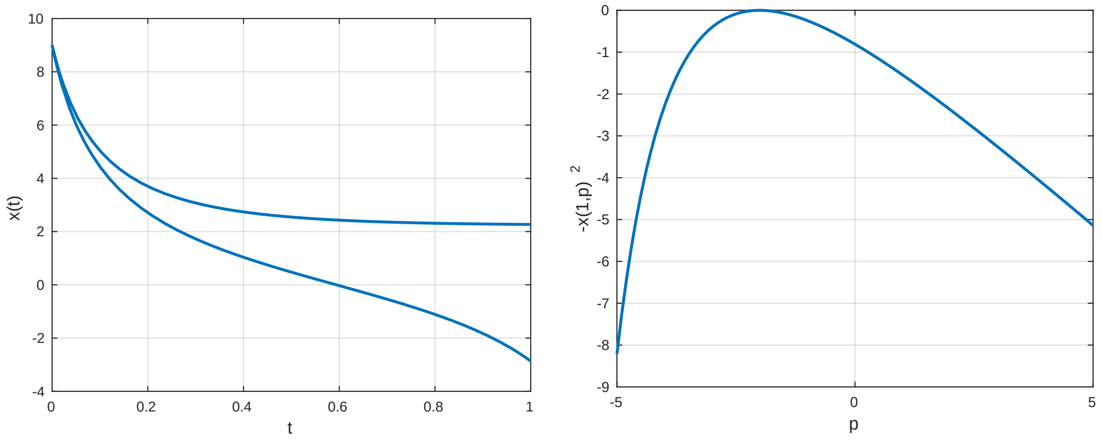


Figure 6.1: Approximation of the upper and lower state bounds of $x(t)$ (left) and the concave objective function (right) over the domain of p of Problem (6.2). The lower bound in the left plot coincides with the trajectory of the globally optimal solution with $p_0^* = -5$.

Figure 6.2 shows the branch-and-bound tree of the direct single shooting approach in a direct comparison with the multiple shooting approach. Even if the objective function is not linear in this case as assumed in Lemma 6, the additional bounds on the state at the end point improve the lower bounds in the first iterations drastically. Nevertheless, both algorithms converge rapidly.

The problem does not contain any control functions, but as mentioned in the introduction, our theory in Section 4.3 indicates that the direct multiple shooting approach leads to an improved convergence nevertheless. Because the convergence to the globally optimal objective function is rapid with all tested methods, we take a closer look at the details. Figure 6.3 shows the convergence of the lower bound to the global solution. For $N = 2$ and $N = 3$, we observe slightly worse lower bounds and an additional iteration. This is explained by the fact that both

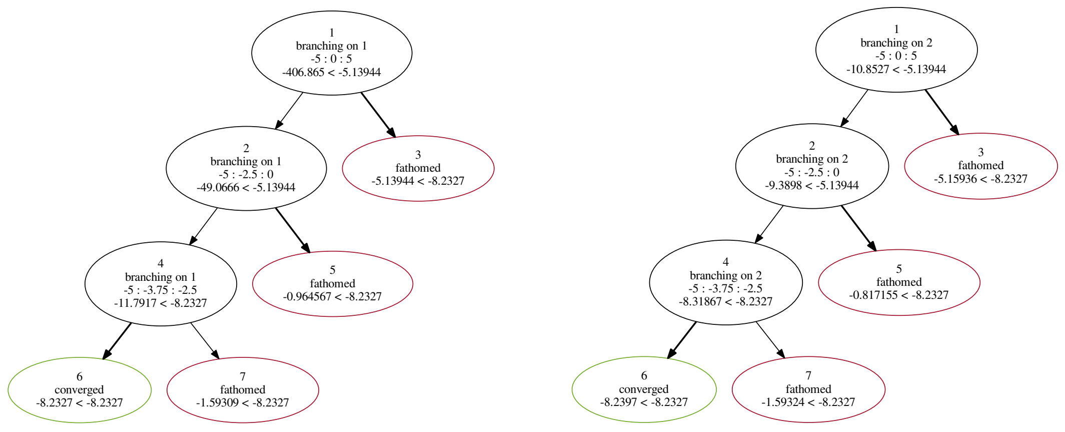


Figure 6.2: Branch-and-bound trees of the global optimal control algorithm for Problem (6.2) using the single shooting approach (left) in comparison to the two point multiple shooting method (right). We notice the same number of iterations and the rapid convergence at the end, but the included state bounds in the two point multiple shooting method improve the first lower bounds drastically.

need one iteration more than the rest. The reason is that in this test example, the validated integrator performs a single step from t_0 to t_N. Therefore, we enforce additional stops to the validated integrator at the multiple shooting nodes for $N \geq 2$ and this may lead to slightly worse relaxations as discussed in Section 4.3.2 and addressed in Corollary 18. The positive effect of using more multiple shooting nodes seem to counteract this, such that for $N \geq 4$ only three iterations are needed and we obtain better bounds in each iteration for an increasing amount of multiple shooting nodes.

Because the convergence behavior itself is similar between the problems, the problem gives an indication of the computational overhead when using an increasing number of multiple shooting intervals. This overhead is mainly due to larger, but sparse NLP problems. Table 6.2 shows the computational times for an increasing amount of multiple shooting nodes.

N	0	1	2	3	4	5	6
time [s]	0.2292	0.1212	0.1865	0.2043	0.1772	0.1902	0.1995

Table 6.2: Computational time for the global optimization of Problem (6.2) with an increasing amount of multiple shooting nodes and 0 indicating single shooting usage.

We notice that for this problem the computational overhead of our algorithm is quite small in comparison to single-shooting-based αBB and due to the efficient handling of the multiple shooting structure, we expect this to be the case for larger problems as well and reinvestigate this for the larger problems in the following sections.

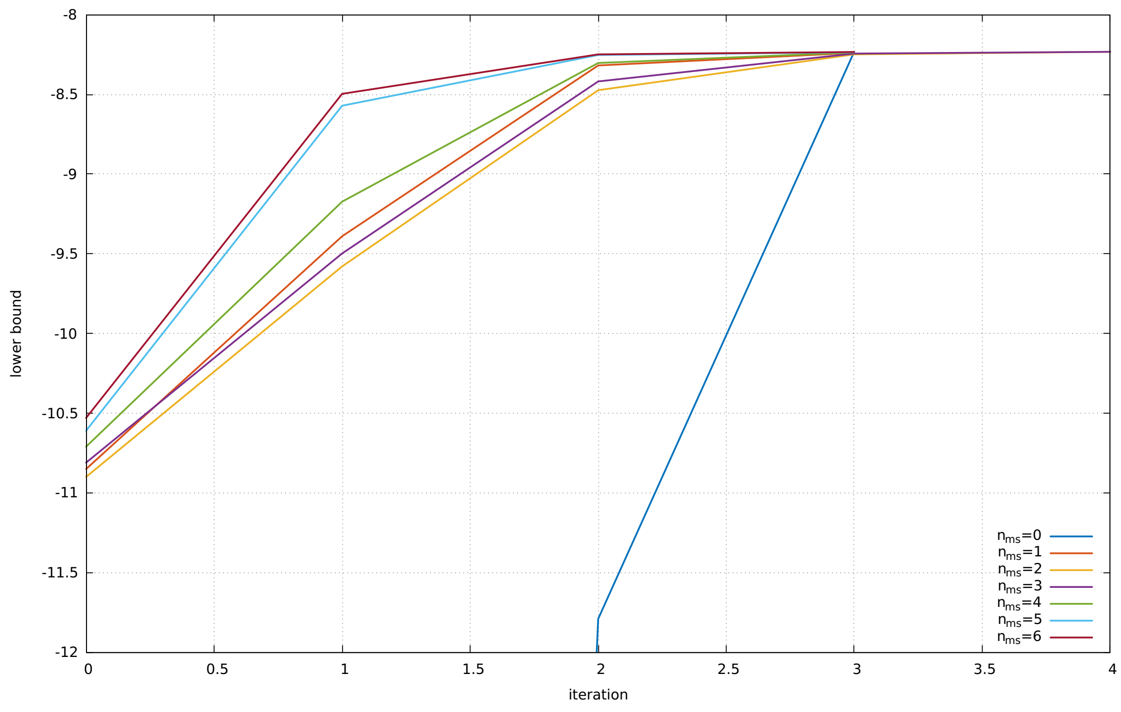


Figure 6.3: Convergence of the lower bound of the objective function over several iterations for Problem (6.2). The lower bound for the single shooting approach is too bad during the first two iterations to plot them in a reasonable way together with the rest, compare the branch-and-bound trees in Figure 6.2 for the corresponding values.

Negative Resistance Circuit

Another test problem from the literature containing only control values, but having a higher dimension than the first problem is the model of a small circuit described in [Kha02] and considered for global optimal control in [Sco12]:

$$\min_{p} \quad x_0(t_N) \tag{6.3a}$$

$$\text{s.t.} \quad \dot{x}_0(t) = p_0 x_1(t) \tag{6.3b}$$

$$\dot{x}_1(t) = -p_1\left(x_0(t) - x_1(t) + \frac{x_1(t)^3}{3}\right) \tag{6.3c}$$

$$t \in [t_0, t_N] = [0, 5] \tag{6.3d}$$

$$p_0 \in [0.01, 0.5], p_1 \in [0.01, 0.5]\,. \tag{6.3e}$$

The global solution is at $(p_0, p_1) = (0.5, 0.5)$ as shown in Figure 6.4 and an additional local solution is reported at $(p_0, p_1) = (0.01, 0.01)$. This problem is interesting as it already poses quite a challenge to our validated integrators on the initial domain. Whereas in the single-shooting-based approach this results in several nodes without a valid lower bound as seen in Figure 6.5, in the direct multiple-shooting-based global optimal control algorithm, we have to branch several times prior to starting Algorithm 5 to obtain validated bounds on all additional multiple shooting variables. This results in 11 root nodes till we are able to obtain state bounds between $[t_0, t_N]$. This special branch-and-bound structure is plotted in Figure 6.6.

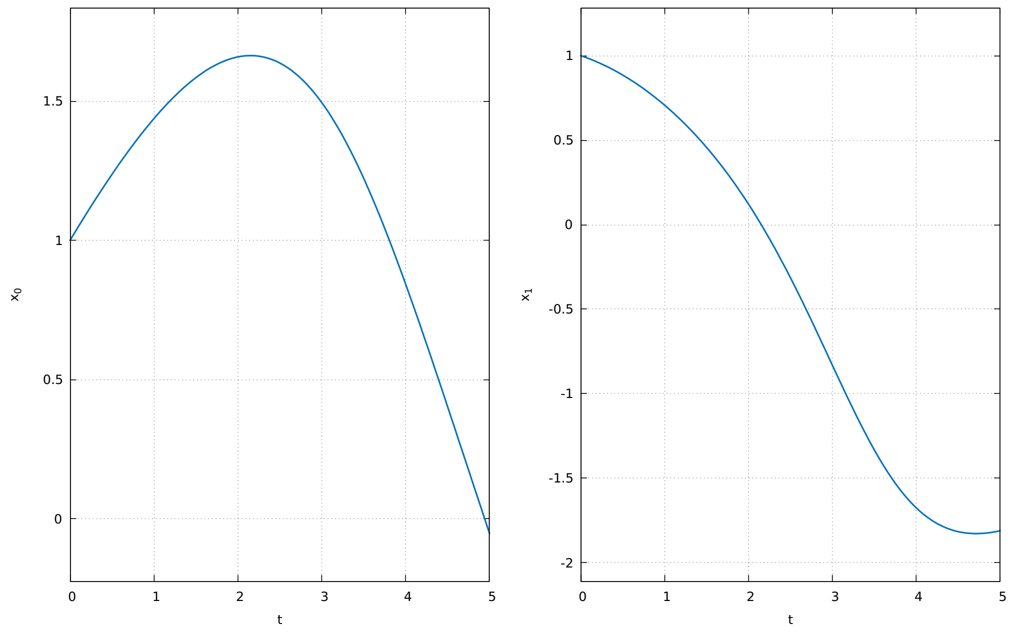


Figure 6.4: Trajectories of the global optimal solution of Problem (6.3) at $(p_0, p_1) = (0.5, 0.5)$.

The computational runtimes of the different algorithms are once more very similar, as shown in Table 6.3, but we notice that the pure single-shooting-based algorithm already performs far worse, because the state bounds at the final time are not included, resulting in a worse relaxation and thus more nodes during the branch-and-bound. Therefore, we focus our comparison on the difference between the direct multiple shooting approach with only two nodes, at t_0 and t_N and the effect of adding more nodes in between. For an increasing amount of multiple shooting nodes, the same effects as for Problem (6.2) are observed.

N	0	1	2	3	4	5	6
time [s]	43.49	8.88	9.38	8.62	8.81	8.16	7.54
iterations	14	4	6	4	5	5	4
time / iteration [s]	3.11	2.22	1.56	2.16	1.76	1.63	1.89
number of nodes	29	29	33	29	31	29	29
time / node [s]	1.50	0.31	0.28	0.30	0.28	0.28	0.26

Table 6.3: Computational time for the global optimization of Problem (6.3) with an increasing amount of multiple shooting nodes and 0 indicating single shooting usage.

Figure 6.7 shows the bounds on the states and the second-order interval sensitivities for an increasing amount of multiple shooting nodes. Whereas the state bound naturally stay the same, we notice the constant initial values of each local sensitivity improves the absolute value of each interval significantly as exploited in Theorem 17.

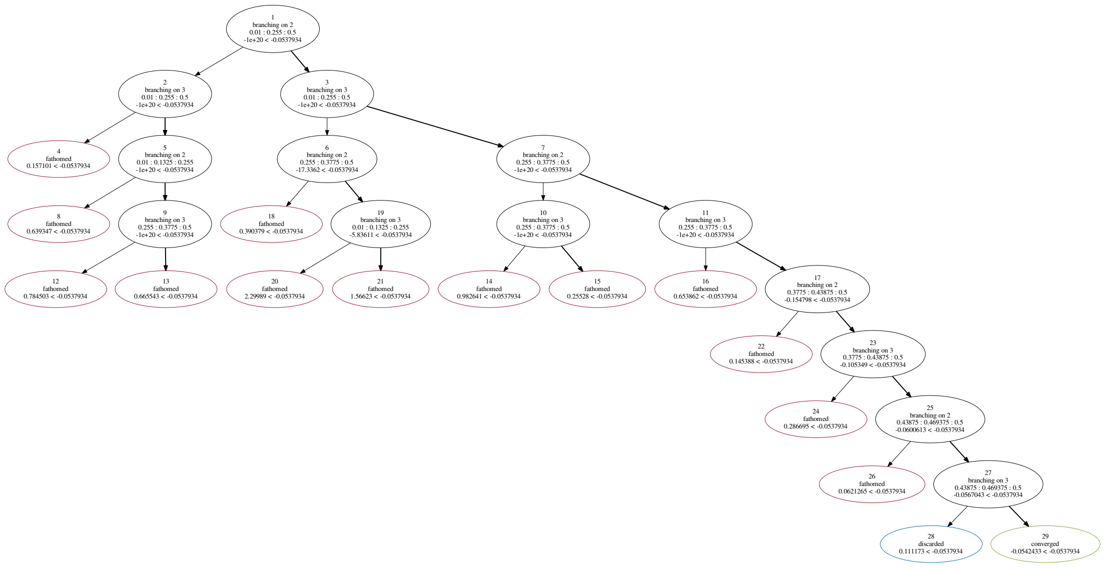


Figure 6.5: The branch-and-bound tree of global optimal control algorithm for Problem (6.3) using the single shooting approach without an initial branching. This is not necessary in this case, because all free variables are already bounded. Nevertheless no valid lower bound is obtained on the first few nodes, because the validated integrator fails to integrate the system and therefore, fails to calculate valid α necessary for the underestimator.

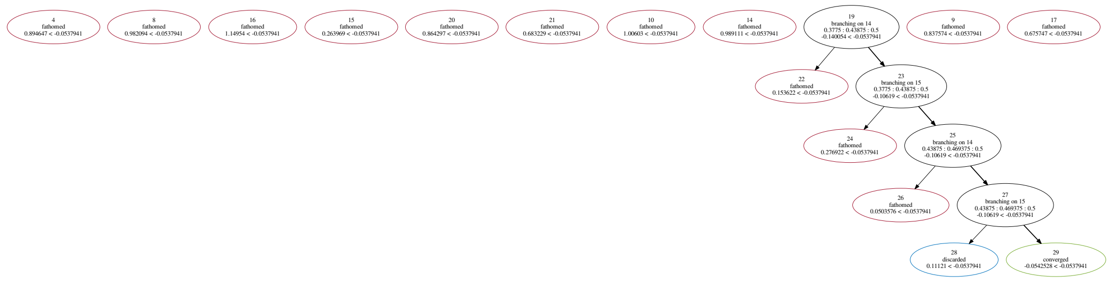


Figure 6.6: The branch-and-bound tree of global optimal control algorithm for Problem (6.3) using the direct multiple-shooting-based method with equidistant multiple shooting intervals. Because there are no bounds given on $s_1, \dots s_6$, we have to integrate the states at least once to obtain validated bounds on the multiple shooting variables. This fails for the initial domain. Therefore, we have to branch multiple times before starting with the core algorithm. This results in the multiple root nodes visualized above, from which all but one can be fathomed directly.

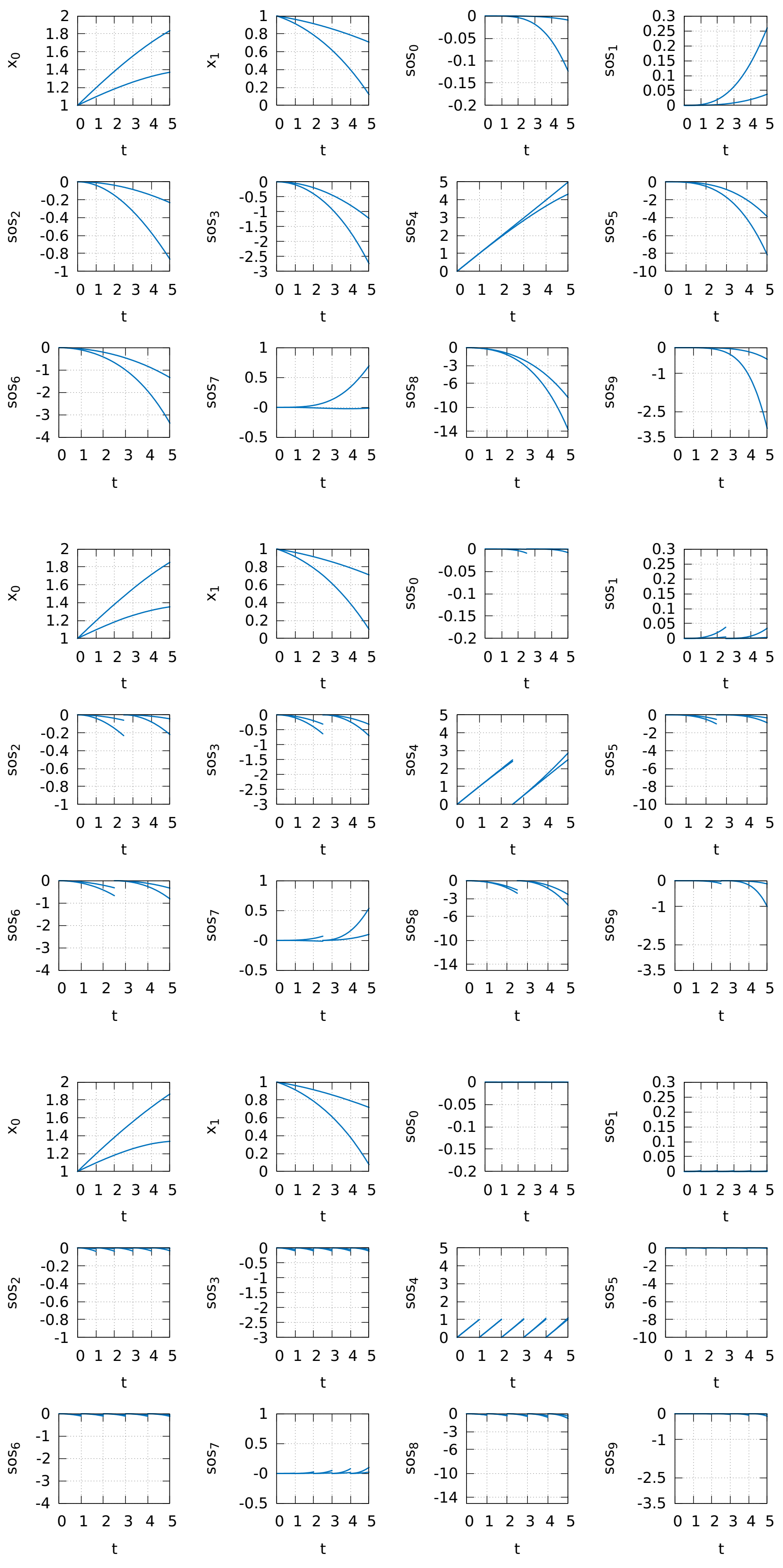


Figure 6.7: Bounds on the states and second-order variational differential equations for an increasing number of multiple shooting intervals $N \in \{1, 2, 5\}$.

6.2.2 Problems with Control Functions

Singular Control Problem

For a first comparison between the direct single shooting approach with the multiple-shooting-based global optimal control algorithm on problems containing control functions and not only control values, we use a common example from the literature that contains a single control function. It is commonly referred to as “singular control problem” and considered for global optimal control in [Luu90], [EF00a], [CL04] and [SB06].
One special property of the problem is the Lagrange type objective function.

$$
\begin{aligned}
\min_{u} \quad & \int_{t_0}^{t_N} x_0(t)^2 + x_1(t)^2 + 0.0005\left(x_1(t) + 16t - 8 - 0.1x_2(t)u(t)^2\right)^2 \,\mathrm{dt} && (6.4a)\\
\text{s.t.} \quad & \dot{x}_0(t) = x_1(t) && (6.4b)\\
& \dot{x}_1(t) = -x_2(t)u(t) + 16t - 8 && (6.4c)\\
& \dot{x}_2(t) = u(t) && (6.4d)\\
& t \in [t_0, t_N] = [0,1], \quad u(t) \in [-4, 10] && (6.4e)\\
& x(t_0) = (0.0, -1.0, -\sqrt{5.0})^T \,. && (6.4f)
\end{aligned}
$$

Introducing the Lagrange objective function as additional state leads to

$$
\begin{aligned}
\min_{u} \quad & x_3(t_N) && (6.5a)\\
\text{s.t.} \quad & \dot{x}_0(t) = x_1(t) && (6.5b)\\
& \dot{x}_1(t) = -x_2(t)u(t) + 16t - 8 && (6.5c)\\
& \dot{x}_2(t) = u(t) && (6.5d)\\
& \dot{x}_3(t) = x_0(t)^2 + x_1(t)^2 + 0.0005\left(x_1(t) + 16t - 8 - 0.1x_2(t)u(t)^2\right)^2 && (6.5e)\\
& t \in [t_0, t_N] = [0,1], \quad u(t) \in [-4, 10] && (6.5f)\\
& x(t_0) = (0.0, -1.0, -\sqrt{5.0}, 0.0)^T \,. && (6.5g)
\end{aligned}
$$

The trajectories of the global optimal solution for different control discretizations are plotted in Figures 6.8 and 6.9. Furthermore, the results from Table 6.4 coincide with the values reported in [LS07a], verifying our implementation.
To get a numerical indication of the quality of the new relaxations based on direct multiple shooting, we compare direct single shooting with state bounds only (DSS1), as described in Algorithm 3, direct single shooting using α-BB relaxations (DSS2), as described in Algorithm 4, direct multiple shooting with only two nodes (DMS1), as described in Algorithm 5 with $N = 1$ and a higher control discretization, and the full direct multiple shooting with several artificial nodes (DMS2), as described in Algorithm 5 with the multiple shooting discretization N equal to the control discretization.
Finally, we include the values for the direct multiple shooting, obtained using reduced space relaxations heuristic described in Section 4.4.3. To validate the method, we compare not only the global solution identified with the full-space-based result, but also the lower bound of the

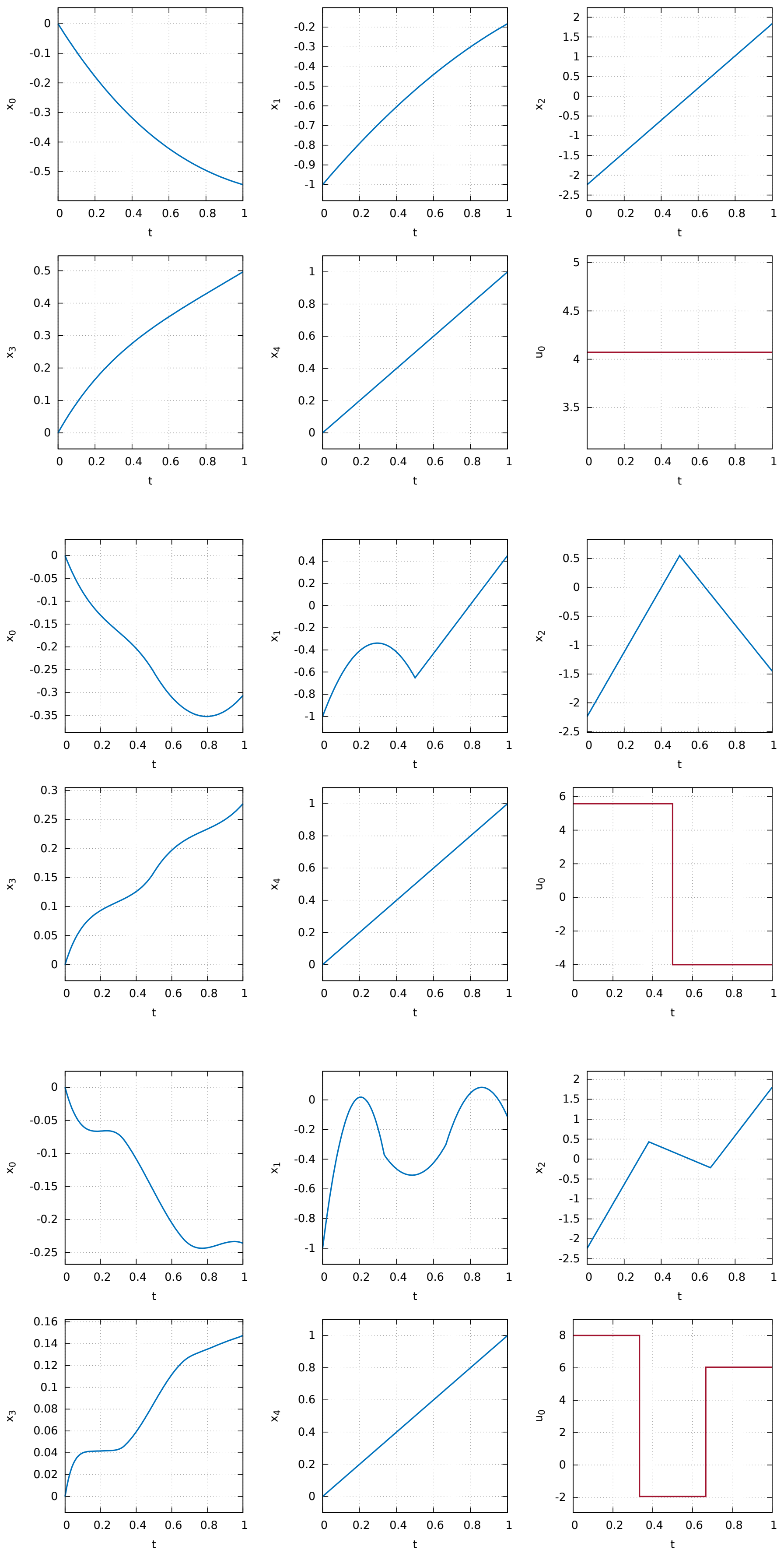


Figure 6.8: Trajectories of the global optimal solution of Problem (6.5) with different control discretizations for $u_{\text{disc}} = 1, \ldots, 3$ (top to bottom).

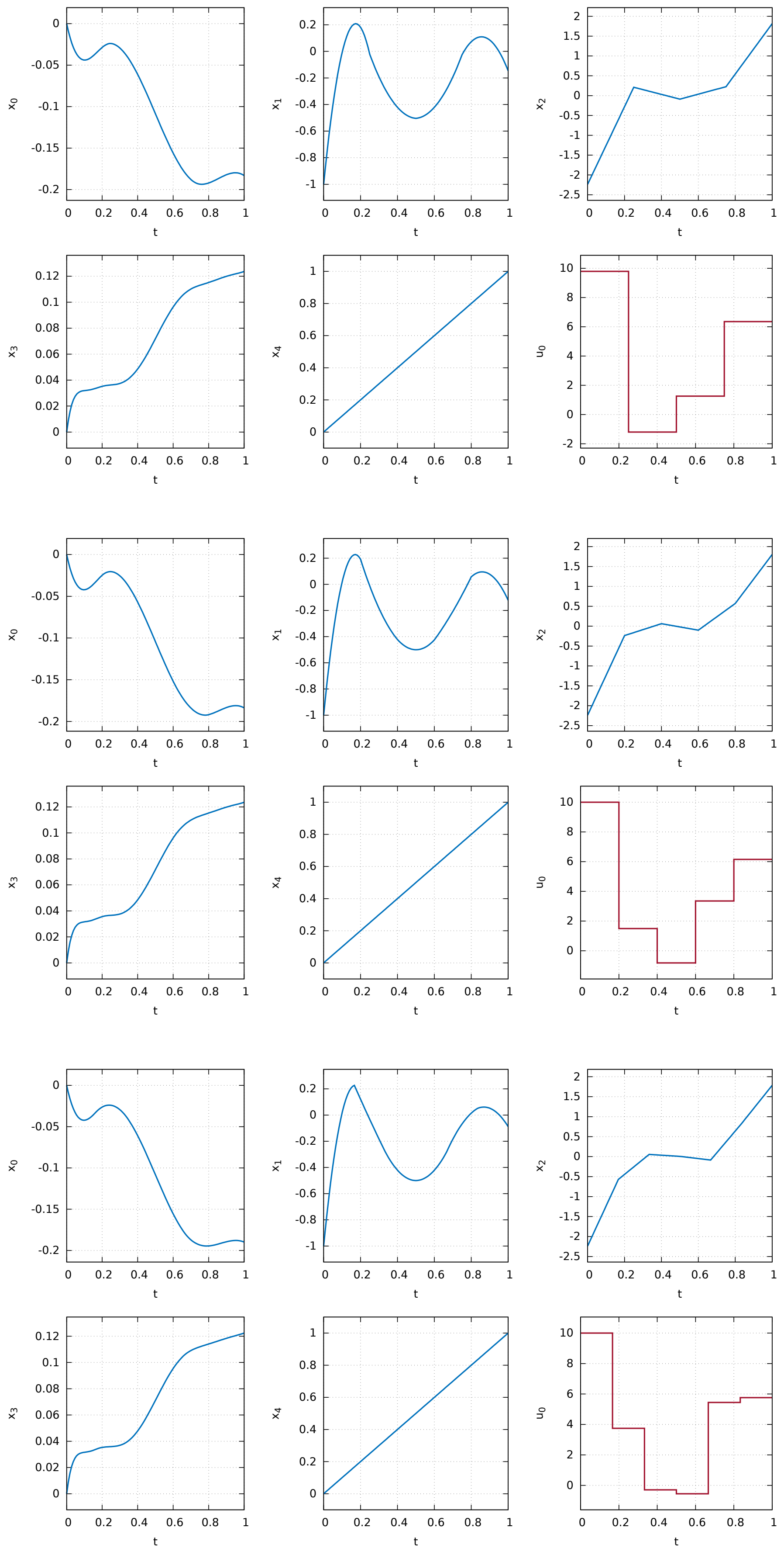


Figure 6.9: Trajectories of the global optimal solution of Problem (6.5) with different control discretizations for $u_{\text{disc}} = 4, \ldots, 6$ (top to bottom).

u_{disc}	Φ	q^{*T}
1	0.4965	(4.0709)
2	0.2771	(5.5748, -4.0000)
3	0.1475	(8.0015, -1.9438, 6.0420)
4	0.1237	(9.7890, -1.1997, 1.2566, 6.3558)
5	0.1236	(9.9998, 1.4939, -0.8144, 3.3540, 6.1514)
6	0.1223	(9.9998, 3.7475, -0.2927, -0.5506, 5.4444, 5.7607)

Table 6.4: Globally optimal objective function value Φ of Problem (6.5) for an increasing number of equidistant control discretization points together with the optimal control parametrization vector q^*.

objective function obtained on each node in the branch-and-bound tree with the corresponding value on the parent node. This test indicates that the parent node has a local minimum and the heuristic failed. For this example, both tests are passed for any tested combination of multiple shooting and control discretization.

Our results in Tables 6.5 and 6.6 show a significant runtime improvement of the direct multiple shooting method presented over the αBB-based direct single shooting method. We abort the iterations at 10000s and mark the corresponding values with "-" if another method is able to obtain the result in the given time frame.

In comparison with the runtimes reported in [LS07a] for DSS1 with a constraint propagation and a different parameter set for the validated integrator, including the order of the Taylor expansion, we note slightly more iterations. Nevertheless, it outperforms the αBB-based methods in this example for the given control discretizations. Although the method needs significantly more iterations, each iteration itself involves just a validated integration of the state bounds. Compared to the αBB-based methods, we save the computational effort of validated integration of the variational differential equations, solving the relaxed NLP to obtain the lower bound and at least periodically solving the original NLP to guarantee the best possible upper bound. This leads to the lowest computational time per node of the methods compared here and this is the reason that in this particular example the improved convex relaxations do not outweigh the additional computational overhead of the αBB relaxation.

Another observation is the difference of DMS1 and DMS2 for one control discretization that are equivalent in theory and the slightly worse performance for two control discretizations. The reason for this effect is that our DMS2 implementation is based on the full space interval sensitivities even on the first multiple shooting interval $[t_0, t_1]$ with fixed initial value s_0. Although the corresponding αBB terms vanish in the under- and overestimations of the matching conditions, using Gershgorin's theorem nevertheless results in worse α_0^q and thus looser constraints in a direct comparison. For more control discretizations, the influence of the first interval becomes smaller and the theoretical advantages of the direct multiple shooting algorithm with additional multiple shooting nodes dominate. As expected, we note the same results for DMS1 and DMSR with $u_{\text{disc}} = 1$.

Comparing DSS2, DMS1 and DMS2 for a higher number of control discretizations, we first notice that the difference between DSS2 and DMS1 becomes smaller for an increasing q_{disc}.

	single shooting			
control	DSS1		DSS2	
intervals	iterations	time [s]	iterations	time [s]
1	9	0.03	1	0.44
2	104	0.29	62	61.87
3	2312	12	1004	2702
4	40319	343	13904	84161
5	674102	18867	-	-
6	-	-	-	-

Table 6.5: Numerical results for different control discretizations of Problem (6.5) using two direct single-shooting-based methods.

	multiple shooting					
control	DMS1		DMS2		DMSR	
intervals	iterations	time [s]	iterations	time [s]	iterations	time [s]
1	0	0.08	0	0.49	0	0.05
2	59	66.6	63	121.6	25	2.28
3	966	2844	782	2196	109	13.4
4	13183	81937	5902	22650	348	51.2
5	-	-	-	-	1144	223
6	-	-	-	-	3742	986

Table 6.6: Numerical results for different control discretizations of Problem (6.5) using the direct multiple-shooting-based approach with a different number of multiple shooting nodes and compared to the reduced space heuristic.

We observe that including the state bound at the final time t_N helps to improve the early relaxations, but at some point the αBB relaxations are tighter and both methods result in the same relaxation for those domains as shown in Lemma 6. This coincides with the experience that αBB relaxations are potentially quite bad over large domains, but have a quadratic convergence behavior near the solution. In contrast to this, the difference of both, DSS2 and DMS1, to our novel multiple shooting algorithm DMS2 is increasing with more control discretizations and multiple shooting nodes. Our approach takes less than 50% of iterations and almost a magnitude less computational time even compared to DMS1. Taking less iterations validates our theoretical predictions from Chapter 4 regarding the tighter relaxations of our direct multiple shooting algorithm and reduce the computational effort.

Regarding the computational effort, for finer control discretizations, we expect this gap to widen further. We notice that a significant amount of the overall computational time is spent on the validated integration of the second-order variational differential equations. We remark that for

$$n_u + n_p \geq n_x + n_q + n_p \tag{6.6}$$

the first- and second-order interval sensitivities become larger for DSS1 and DMS1 compared to DMS2 with an equivalent multiple shooting and control discretization grid. This has not only a positive impact on the convex relaxations as seen in Section 4.3.3, but also reduces the time to update α on each node.

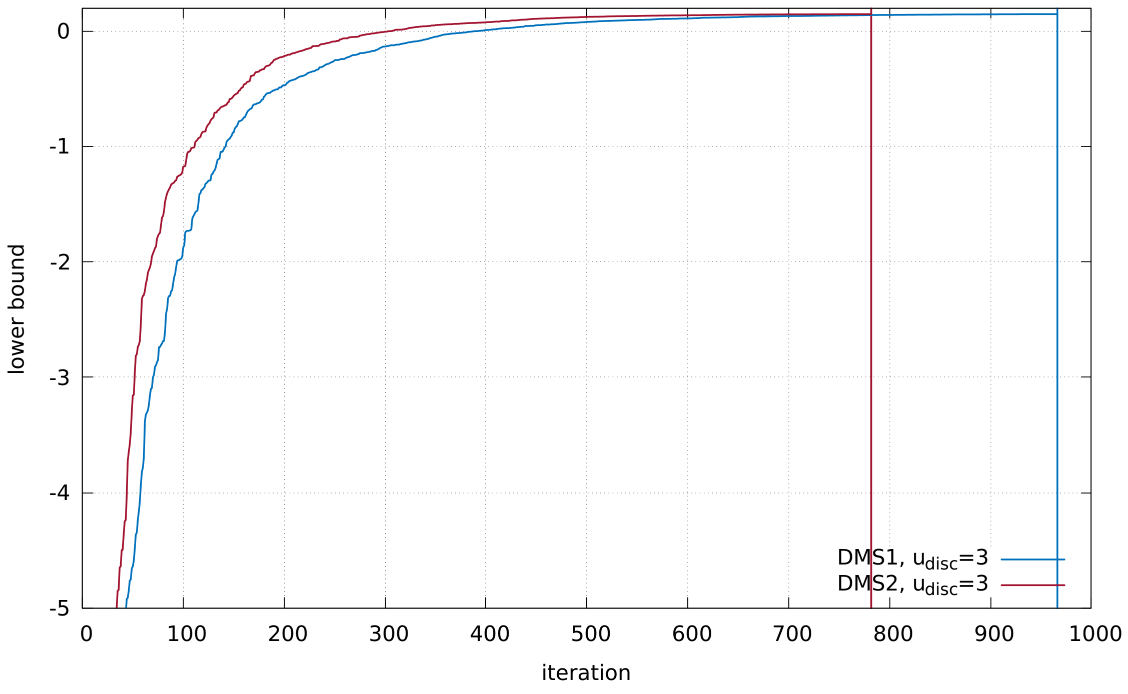
Updating α based on the second-order interval sensitivities is significantly more expensive in global optimal control compared to calculating an interval Hessian in non-dynamic global optimization. Therefore, we investigate if we are able to speed up the runtime by skipping updating α and instead only update the state bounds. This is a common method in αBB literature, because sufficiently large α values remain sufficiently large and the under- and overestimators are getting better in each iteration due to the tightened variables bounds. In theory this may speed up the overall runtime at the expense of the number of iterations. To obtain the results shown in Tables 6.7 and 6.8, we postponed the α updates an increasing amount of times, depending on the number of branches already performed on a certain node, up to a maximum number of ten update skips. To differentiate in the notation, we call them DSS1s, DMS1s, DMS2s and DMSRs. Naturally, this is not an option for DSS1, as no αBB relaxations are performed in this case.

	single shooting	
control	DSS2s	
intervals	iterations	time [s]
1	2	0.92
2	102	50.2
3	1521	1213
4	25483	34435
5	-	-
6	-	-

Table 6.7: Numerical results for different control discretizations of Problem (6.5) obtained while delaying the α updates, resulting in faster runtimes at the expense of more iterations.

Figures 6.10 and 6.11 visualize the significantly faster convergence rate of our direct multiple shooting algorithm with three and four multiple shooting intervals compared to the two point multiple shooting approach in both the original form and the variant with skipped α updates.

To compare the quality of the relaxations of Problem (6.5), Tables 6.9 and 6.10 provide more insight by providing the lower bounds obtained for the objective functions for various different domains, including the original one at the root node and various in an ϵ-box around the globally optimal solution obtained before and cut off at the original bound if necessary. The table shows that the additional computational effort using αBB leads indeed to better lower bounds. Furthermore, we observe as expected by our theoretical results that the bounds obtained by the DMS2 method are significantly better than the bounds obtained through single shooting



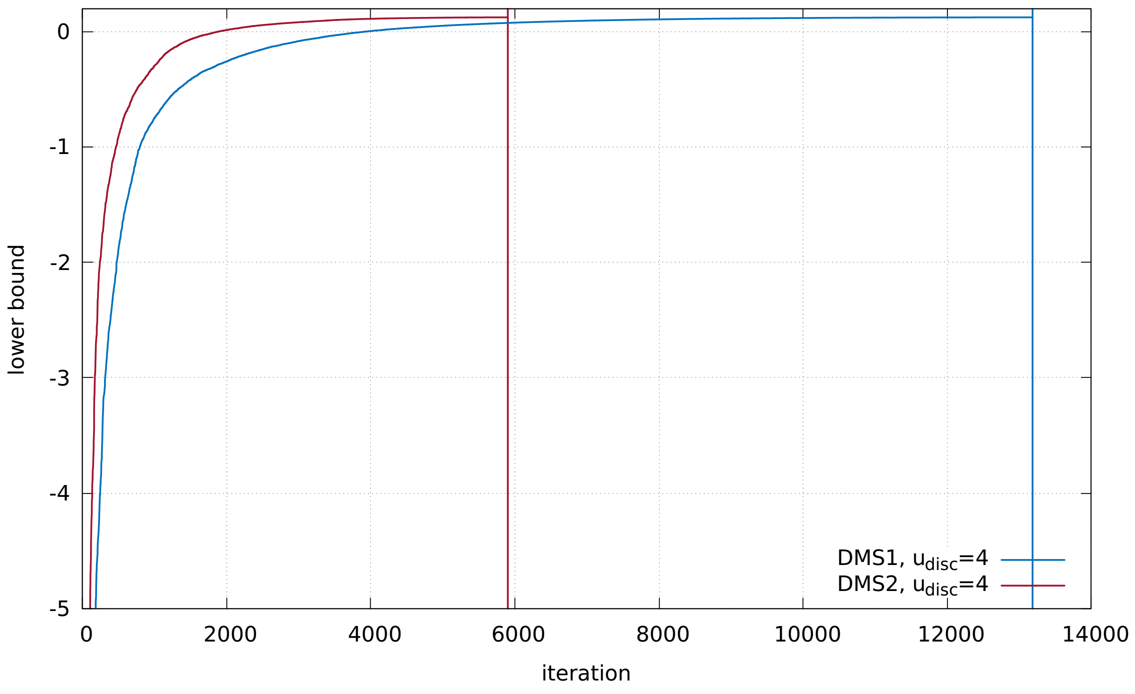
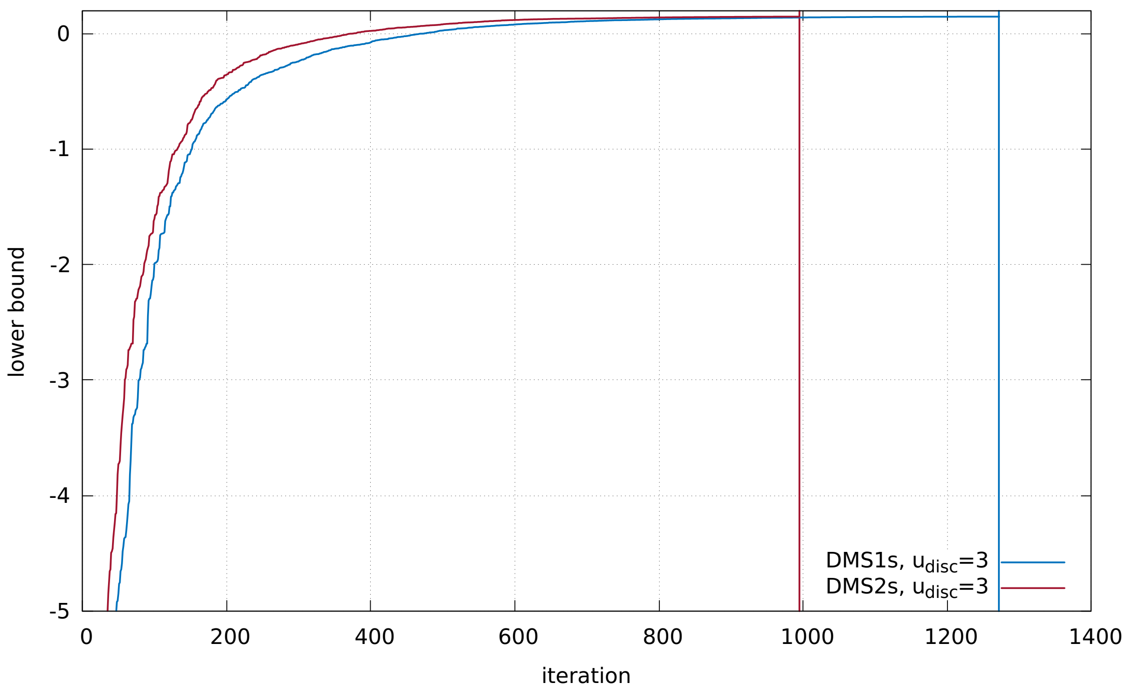


Figure 6.10: A direct comparison of the convergence of the lower bound for the global optimal solution for Problem (6.5) with three (top) and four (bottom) equidistant control discretizations between DMS1 and DMS2.



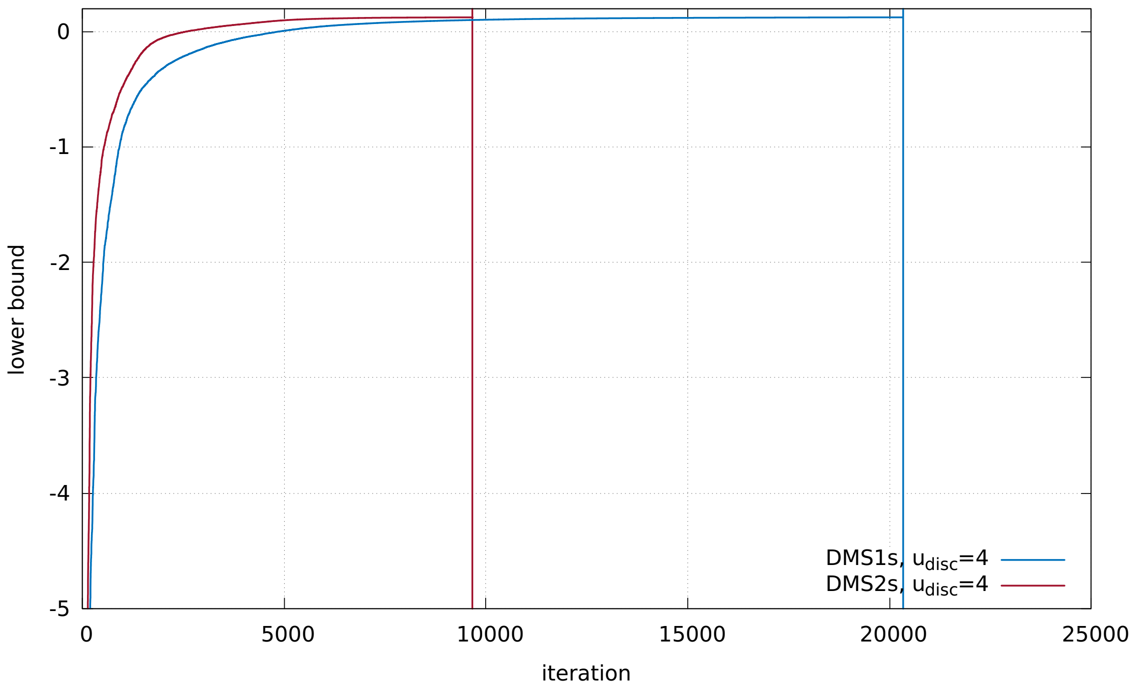


Figure 6.11: A direct comparison of the convergence of the lower bound for the global optimal solution for Problem (6.5) with three (top) and four (bottom) control discretizations between DMS1s and DMS2s.

	multiple shooting					
control	DMS1s		DMS2s		DMSRs	
intervals	iterations	time [s]	iterations	time [s]	iterations	time [s]
1	0	0.05	0	0.46	0	0.08
2	73	28.9	73	46.4	31	2.16
3	1272	927	995	729.9	167	14.0
4	20330	22145	9671	8925	596	55.2
5	-	-	87028	98908	1716	179
6	-	-	-	-	6984	980

Table 6.8: Numerical results for different control discretizations of Problem (6.5) obtained while delaying the α updates, resulting in faster runtimes at the expense of more iterations.

and the two point multiple shooting approach.

		single shooting	
control		DSS1	DSS2
intervals	componentwise domain	lower bound	lower bound
1	$[-4.0, 10.0]$	-10.8183	-0.735976
	$[-4.0, -3.95]$	128.981	128.981
	$[-9.95, 10.0]$	138.949	138.949
	$q_i^* + [-1.0; 1.0]$	0.236938	0.496545
	$q_i^* + [-0.1; 0.1]$	0.496285	0.496545
	$q_i^* + [-0.01; 0.01]$	0.496544	0.496545
3	$[-4.0, 10.0]$	-187.834	-906.238
	$[-4.0, -3.95]$	128.969	128.981
	$[-9.95, 10.0]$	138.936	138.949
	$q_i^* + [-1.0; 1.0]$	-0.319458	-0.767086
	$q_i^* + [-0.1; 0.1]$	0.145153	0.146103
	$q_i^* + [-0.01; 0.01]$	0.147455	0.147468
6	$[-4.0, 10.0]$	-252.515	-993.86
	$[-4.0, -3.95]$	128.966	128.981
	$[-9.95, 10.0]$	138.932	138.949
	$q_i^* + [-1.0; 1.0]$	-0.250874	-0.605863
	$q_i^* + [-0.1; 0.1]$	0.119893	0.120369
	$q_i^* + [-0.01; 0.01]$	0.122351	0.122361

Table 6.9: Lower bounds on the solution of Problem (6.5) at the certain nodes of the branch-and-bound tree.

		multiple shooting		
control		DMS1	DMS2	DMSR
intervals	componentwise domain	lower bound	lower bound	lower bounds
1	$[-4.0, 10.0]$	0.496546	0.496546	0.496546
	$[-4.0, -3.95]$	128.981	128.981	128.981
	$[-9.95, 10.0]$	138.949	138.949	138.949
	$q_i^* + [-1.0; 1.0]$	0.496546	0.496546	0.496546
	$q_i^* + [-0.1; 0.1]$	0.496546	0.496546	0.496546
	$q_i^* + [-0.01; 0.01]$	0.496545	0.496545	0.496545
3	$[-4.0, 10.0]$	-187.834	-187.834	-55.3939
	$[-4.0, -3.95]$	128.981	128.981	128.981
	$[-9.95, 10.0]$	138.949	138.949	138.949
	$q_i^* + [-1.0; 1.0]$	-0.319456	-0.312031	0.117835
	$q_i^* + [-0.1; 0.1]$	0.146104	0.146709	0.147482
	$q_i^* + [-0.01; 0.01]$	0.147469	0.147474	0.147479
6	$[-4.0, 10.0]$	-252.515	-169.889	-17.7529
	$[-4.0, -3.95]$	128.981	128.981	128.981
	$[-9.95, 10.0]$	138.949	138.949	138.949
	$q_i^* + [-1.0; 1.0]$	-0.250872	-0.0534457	0.117001
	$q_i^* + [-0.1; 0.1]$	0.12037	0.12185	0.122393
	$q_i^* + [-0.01; 0.01]$	0.122362	0.122381	0.122382

Table 6.10: Lower bounds on the solution of Problem (6.5) at the certain nodes of the branch-and-bound tree.

The last numerical result based on this test example from the literature is the observation of the so-called cluster effect that is described in [DK94] with potential remedies given in [SN04, WSB14], where the global optimal solution is identified quite early, but it takes many branches around this solution to verify the solution up to the desired accuracy. Using only two control discretizations, we are able to visualize this effect, as shown in Figure 6.12.

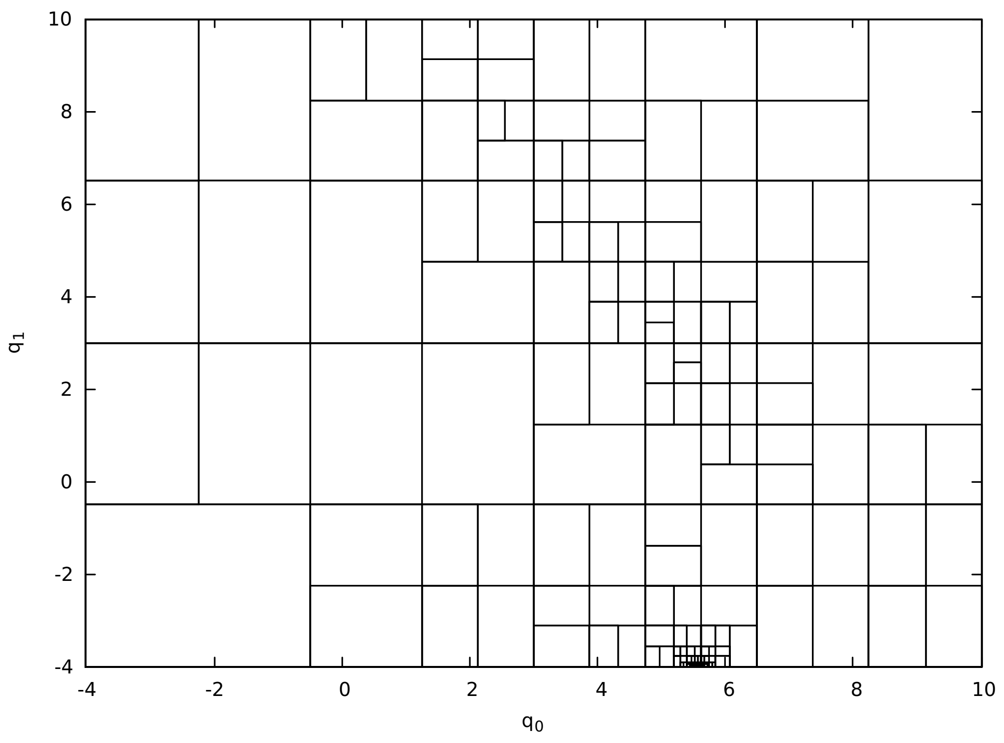


Figure 6.12: Visualization of the cluster effect on Problem (6.5). The global solution is identified early, but to achieve the desired accuracy, the algorithm has to branch a large number of times in the vicinity of the global solution.

Oil Shale Pyrolysis Problem

The last test problem from the global optimal control literature considered in this thesis is an oil shale pyrolysis model introduced in [WY77] and subsequently considered for global optimization in [Luu90] and others. [EF00a] reports eight known minima. We use the normalized formulation from [LS07a]. With $u(t) := \frac{1}{u_{\text{temp}}}$ being the inverse of the controlled temperature $u_{\text{temp}} \in [698.15, 748.15]$, we obtain

$$\min_{u} \quad -x_1(t_N)^2 \tag{6.7a}$$

$$\text{s.t.} \quad \dot{x}_0(t) = -k_0 x_0(t) - (k_2 + k_3 + k_4) x_0(t) x_1(t) \tag{6.7b}$$

$$\dot{x}_1(t) = k_0 x_0(t) - k_1 x_1(t) + k_2 x_0(t) x_1(t) \tag{6.7c}$$

$$k_i = a_i e^{-u(t)\frac{b_i}{R}} \quad \forall i \in \{1, \dots, 5\} \tag{6.7d}$$

$$t \in [t_0, t_N] = [0, 10], \quad u(t) \in [698.15/748.15, 1] \tag{6.7e}$$

$$x(t_0) = (1.0, 0.0)^T \tag{6.7f}$$

with a_i and b_i as stated in Table 6.11 and $R = 1.9872e - 3$.

a_1	a_2	a_3	a_4	a_5	b_1	b_2	b_3	b_4	b_5
8.86	24.25	23.67	18.75	20.70	20.3	37.4	33.8	28.2	31.0

Table 6.11: Model parameters from [WY77] for Problem (6.7).

Algorithm DMS1 is for linear objective functions always as good as DSS2 as shown in Lemma 6 and we have seen in the numerical examples so far that with an increasing amount of control discretizations their convergence behavior is very similar. Therefore, we restrict further comparisons to the DMS1, DMS2 and DMSR.

In this example, we observe that similar to Problem 6.3, the validated integrator fails to obtain state bounds for the root node. Even with an increased order of the Taylor model, we have 9, 31, 132 and 625 root nodes for $u_{\text{disc}} = 1, \dots, 4$ after branching till every node is bounded. Furthermore, we observe that the validated integrator is not able to obtain a valid α for the whole horizon and therefore, the derivation of an αBB underestimator fails for methods DSS2 and DMS1, whereas the DMS2 algorithm succeeds in calculating a valid underestimator. This effect increases the runtime difference between the different approaches.

Figures 6.13 and 6.14 show the global optimal solution for different control discretizations.

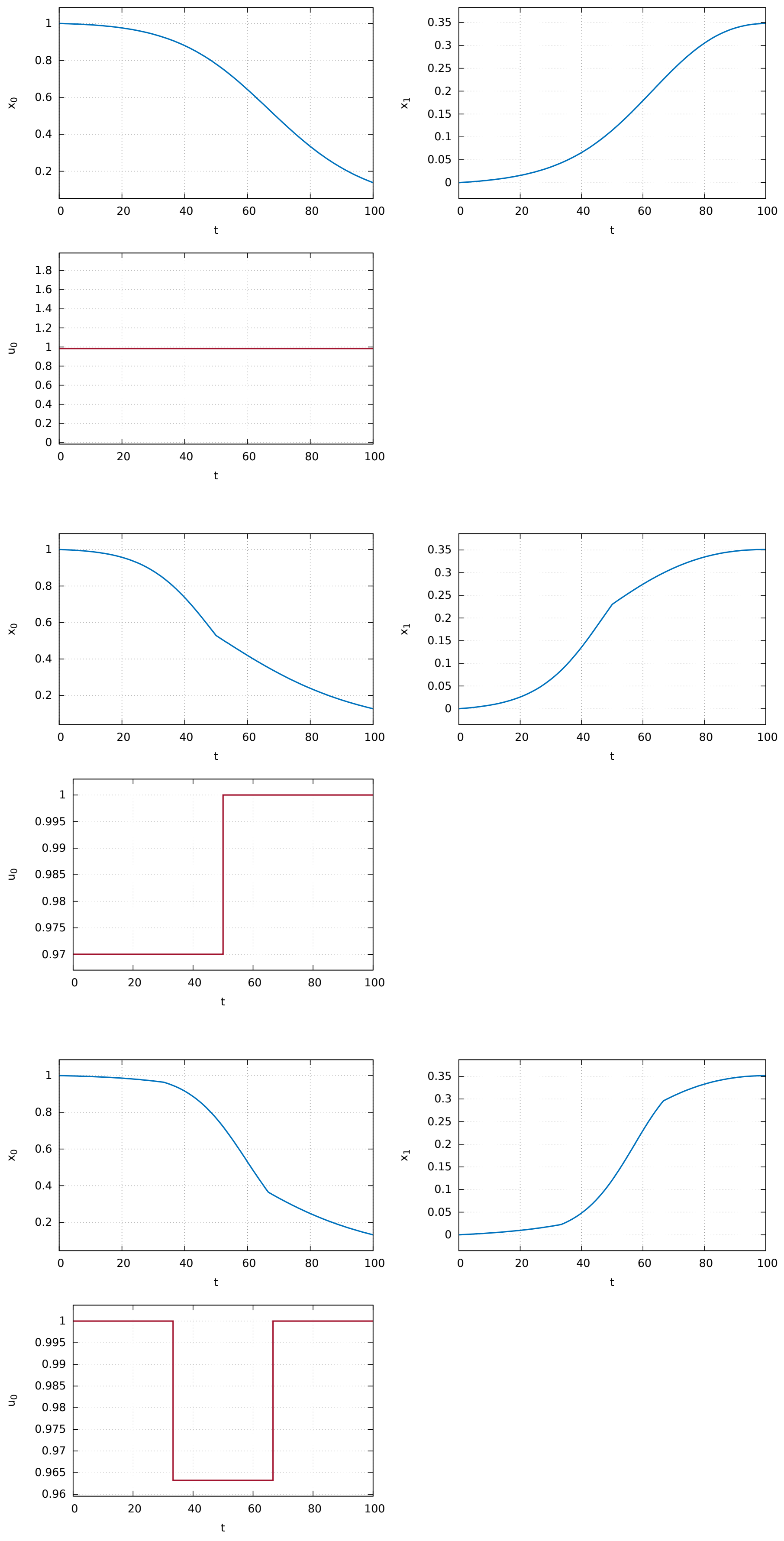


Figure 6.13: Trajectories of the global optimal solution of Problem (6.7) with different control discretizations for $u_{\text{disc}} = 1, \dots, 3$.

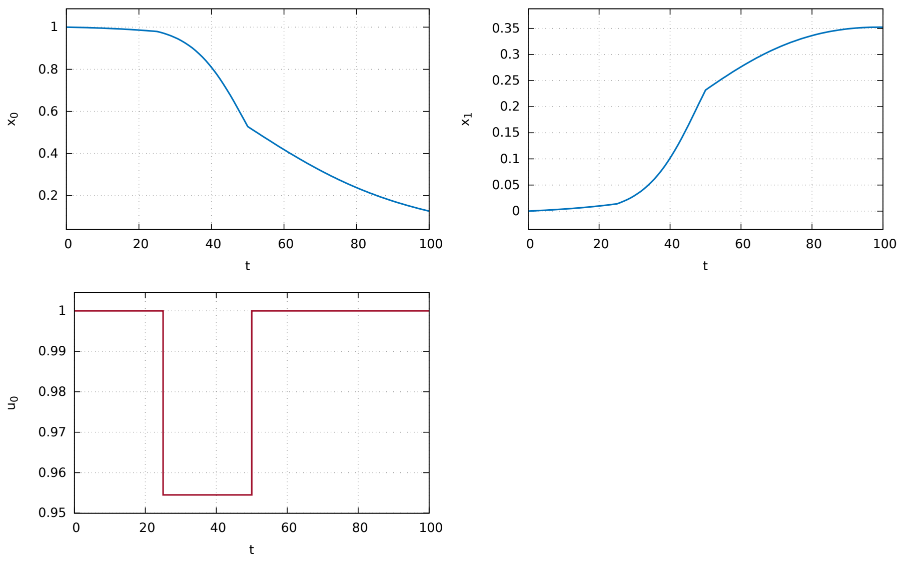


Figure 6.14: Trajectories of the global optimal solution of Problem (6.7) with a different control discretizations of $u_{\text{disc}} = 4$.

6.3 Larger Optimal Control Application

In this section, we present an application where multiple local minima were observed, but that was not tackled in the global optimal control literature so far. In contrast to the other problems from the literature presented in the last section, the following robot model contains two control functions and we use a higher dimensional control discretization to test the limits of our method. Furthermore, it contains a free final time t_N and fixed initial values as well as fixed states at $x(t_N)$, making it a boundary value problem and an additional challenge for the global optimization.

2D Robot Arm

This planar robot arm is fully derived in [Ved11]. The application is interesting for global optimal control, because even for such a small robotic model, the cited diploma thesis pointed out many local minima. Some of these local minima are due to symmetric movements, but a robust identification of the true global optimum is desirable. The challenges regarding this model is a highly nonlinear behavior, a finer control discretization and boundary conditions at the final free time t_N.

$$\begin{aligned}
\min_{t_N,u}\quad & t_N && \text{(6.8a)}\\
\text{s.t.}\quad & \ddot{x}_0(t) = x_2(t) && \text{(6.8b)}\\
& \ddot{x}_1(t) = x_3(t) && \text{(6.8c)}\\
& \ddot{x}_2(t) = B_3(x)[J_3 B_1(x,u) + B_0(x) B_2(x,u)] && \text{(6.8d)}\\
& \ddot{x}_3(t) = B_3(x)[B_0(x) B_1(x,u) + J_2 B_2(x,u)] && \text{(6.8e)}\\
& t \in [t_0, t_N] = [0, t_N] && \text{(6.8f)}\\
& u_0(t) \in [-800.0, 800.0], \quad u_1(t) \in [-800.0, 800.0] && \text{(6.8g)}\\
& t_N \in [0, 1.0] && \text{(6.8h)}
\end{aligned}$$

with

$$\begin{aligned}
B_0(x) &= -k_4 \cos(x_1(t) - x_0(t))\\
B_1(x,u) &= k_4 x_3(t)^2 \sin(x_1(t) - x_0(t)) - k_0 \cos(x_0(t)) + (u_0(t) - u_1(t))\\
B_2(x,u) &= -k_4 x_2(t)^2 \sin(x_1(t) - x_0(t)) - k_1 \cos(x_1(t)) + u_1(t)\\
B_3(x) &= \frac{1}{k_3 - k_2 \cos^2(x_1(t) - x_0(t))}
\end{aligned}$$

and constants

$$k_0 = 470.88, \quad k_1 = 156.96, \quad k_2 = 163.84, \quad k_3 = 291.27 + \frac{1}{900}, \quad k_4 = 12.8$$

$$J_2 = 34.1 + \frac{1}{30}, \quad J_3 = 8.5 + \frac{1}{30}\,.$$

Apart from the two control functions, the end time t_N is free in this application and we choose to minimize the time necessary for a certain maneuver by fixing the initial position and speed, as well corresponding values at the final time:

$$\begin{aligned} x_0(t_0) &= 0 & x_0(t_N) &= \frac{\pi}{3} \\ x_1(t_0) &= 0 & x_1(t_N) &= \frac{\pi}{3} \\ x_2(t_0) &= 0 & x_2(t_N) &= 0 \\ x_3(t_0) &= 0 & x_3(t_N) &= 0\,. \end{aligned}$$

We are not able to obtain validated bounds for the states and the variational differential equations in this case and have to resort to sample the second-order interval sensitivities in order to determine α. This is common technique in αBB-based optimal control literature. We obtain for this approximation $\tilde{H}$ that

$$\tilde{H} \subseteq H\,. \tag{6.9}$$

A different validated integrators may solve this in the future. Nevertheless, we continue to use Gershgorin's circle theorem to over- and underestimate the eigenvalues instead of calculating the eigenvalues of the sampled sensitivites.

Although the problem proves to be too big for the full space approach, we are able to identify the global optimal solution shown in Figure 6.15 for an equidistant piecewise constant control discretization in comparison to two other local minima using our reduced space heuristic presented in Section 4.4.3.

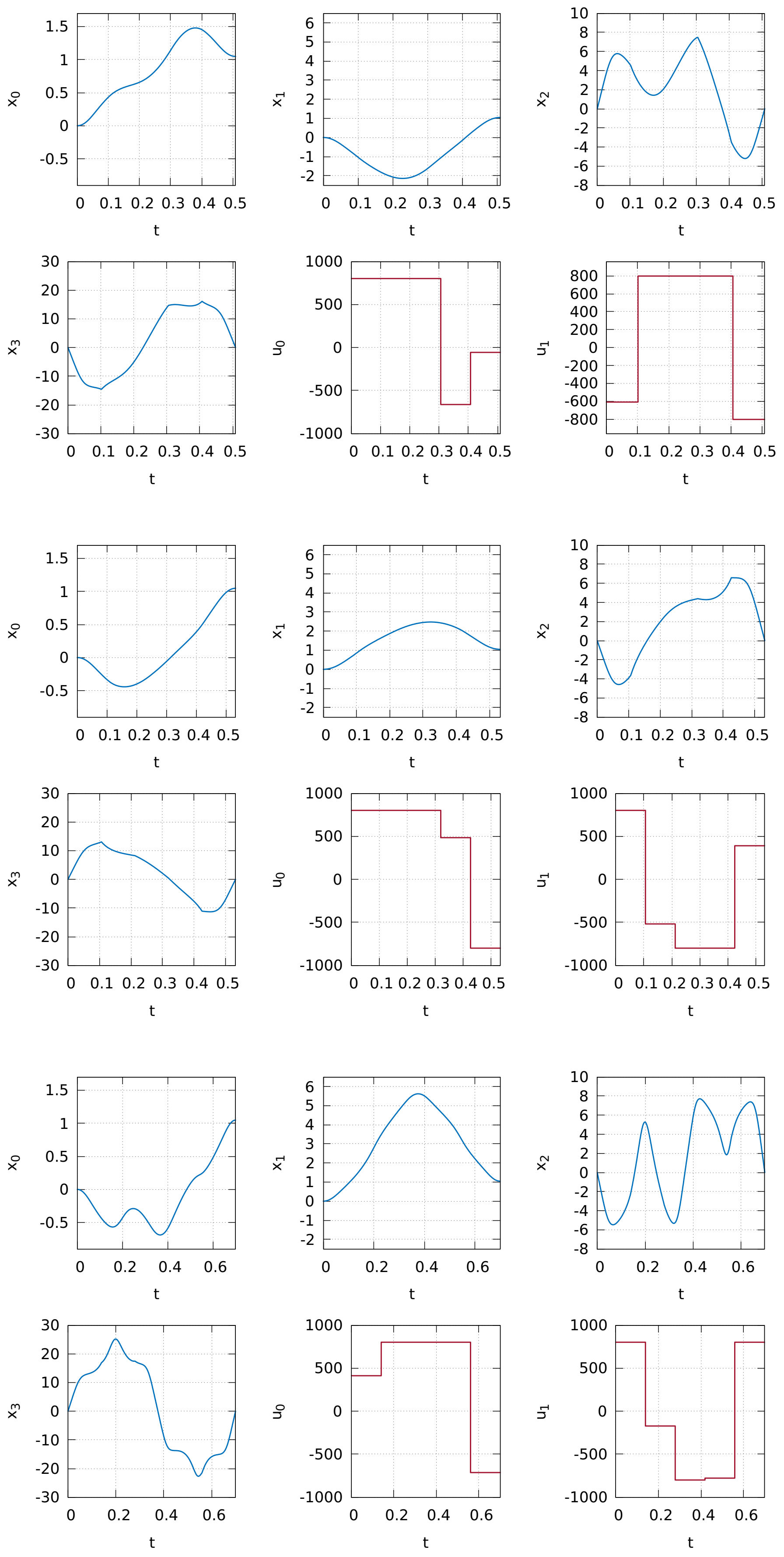


Figure 6.15: Trajectories of the global solution (top) in comparison to two worse local solutions of Problem (6.8) with a control discretization of $u_{\text{disc}} = 5$.

6.4 Global Optimal Control for Problems Containing Integer Decisions

One of our motivations for global optimal control are mixed integer optimal control problems (MIOCP) [Sag06, SRB09]. Regarding global optimal control for MIOCPs, we refer to [CSB05, CSB06, SCM15]. The synergy is to relax the integrality of any integer-valued control functions or control parameters and convexify the problem at the same time and handling both aspects in the same branch-and-bound tree to obtain convex lower bounds for the MIOCP.

Aircraft Model

The F8 aircraft model goes back to [KN03] and to [Sag05] in the optimal control formulation that we use. It is fully described, including source code in several languages, on `mintoc`[Sag, Sag12], which is a benchmark library for MIOCPs. There are multiple local minima reported by different authors, including Schlueter et al in [SEG+13].

The model itself contains three state variables. $x_0(t)$ is the angle of attack in radians, $x_1(t)$ the pitch angle and $x_2(t)$ is the pitch rate in rad/s. The time dependent control function $w(t)$ is the tail deflection angle in radians and is from a discrete set of two decisions.

The problem formulation is the following:

$$\begin{array}{llcl}
\min\limits_{w,t_N} & t_N & & \\
\text{s.t.} & \dot{x}_0(t) & = & -0.877x_0(t)+x_2(t)-0.088x_0(t)x_2(t)+0.47x_0(t)^2 \\
& & & -0.019x_1(t)^2-x_0(t)^2x_2(t)+3.846x_0(t)^3 \\
& & & -0.215w(t)+0.28x_0(t)^2w(t)+0.47x_0(t)w(t)^2+0.63w(t)^3 \\
& \dot{x}_1(t) & = & x_2(t) \\
& \dot{x}_2(t) & = & -4.208x_0(t)-0.396x_2(t)-0.47x_0(t)^2-3.564x_0(t)^3 \\
& & & -20.967w(t)+6.265x_0(t)^2w(t)+46x_0(t)w(t)^2+61.4w(t)^3 \\
& x(0) & = & (0.4655,0,0)^T, \\
& x(t_N) & = & (0,0,0)^T, \\
& w(t) & \in & \{-0.05236, 0.05236\}\,.
\end{array} \tag{6.10}$$

Using the partial outer convexification approach from [Sag05, SRB09], we can obtain a formulation that is linear in the control:

$$
\begin{array}{lrcl}
\min\limits_{w,t_N} & t_N & & \\
\text{s.t.} & \dot{x}_0(t) & = & -0.877x_0(t) + x_2(t) - 0.088x_0(t)x_2(t) + 0.47x_0(t)^2 \\
& & & -0.019x_1(t)^2 - x_0(t)^2x_2(t) + 3.846x_0(t)^3 \\
& & & +0.215\xi - 0.28x_0(t)^2\xi + 0.47x_0(t)\xi^2 - 0.63\xi^3 \\
& & & -\big(0.215\xi - 0.28x_0(t)^2\xi - 0.63\xi^3\big)\,2w(t) \\
& \dot{x}_1(t) & = & x_2(t) \\
& \dot{x}_2(t) & = & -4.208x_0(t) - 0.396x_2(t) - 0.47x_0(t)^2 - 3.564x_0(t)^3 \\
& & & +20.967\xi - 6.265x_0(t)^2\xi + 46x_0(t)\xi^2 - 61.4\xi^3 \\
& & & -\big(20.967\xi - 6.265x_0(t)^2\xi - 61.4\xi^3\big)\,2w(t) \\
& x(0) & = & (0.4655, 0, 0)^T, \\
& x(t_N) & = & (0, 0, 0)^T, \\
& w(t) & \in & \{0, 1\}, \qquad \xi = 0.05236\,.
\end{array}
\tag{6.11}
$$

Due to the usually relatively coarse control discretization in global optimal control, we use this application as a numerical test for a global switching point optimization. Therefore, we split the model into N model stages each with a free end time p_j. For a given integral switching structure of $w(t)$, we obtain

$$
\begin{array}{lrcl}
\min\limits_{p} & \sum_{j=0}^{N} p_j & & \\
\text{s.t.} & \dot{x}_0(t) & = & \big(-0.877x_0(t) + x_2(t) - 0.088x_0(t)x_2(t) + 0.47x_0(t)^2 \\
& & & -0.019x_1(t)^2 - x_0(t)^2x_2(t) + 3.846x_0(t)^3 \\
& & & +0.215\xi - 0.28x_0(t)^2\xi + 0.47x_0(t)\xi^2 - 0.63\xi^3 \\
& & & -\big(0.215\xi - 0.28x_0(t)^2\xi - 0.63\xi^3\big)\,2w(t)\big)p_j \\
& \dot{x}_1(t) & = & x_2(t)p_j \\
& \dot{x}_2(t) & = & \big(-4.208x_0(t) - 0.396x_2(t) - 0.47x_0(t)^2 - 3.564x_0(t)^3 \\
& & & +20.967\xi - 6.265x_0(t)^2\xi + 46x_0(t)\xi^2 - 61.4\xi^3 \\
& & & -\big(20.967\xi - 6.265x_0(t)^2\xi - 61.4\xi^3\big)\,2w(t)\big)p_j\,, \\
& & & \quad \forall t \in [j, j+1),\ j \in \{0, \dots, N-1\} \\
& x(0) & = & (0.4655, 0, 0)^T, \\
& x(N) & = & (0, 0, 0)^T, \\
& w(t) & = & 1\,, \quad \forall t \in [0,1), t \in [2,3), t \in [4,5] \\
& w(t) & = & 0\,, \quad \forall t \in [1,2), t \in [3,4), t \in [5,6] \qquad \xi = 0.05236 \\
& p_j & \in & [0.01, 3.0]\,, \quad \forall j \in \{1, \dots, N\}\,.
\end{array}
\tag{6.12}
$$

Furthermore, we introduce a safeguard at the lower bound of p to prevent numerical instabilities. This small value should still be able to approximate solutions with less switches within the solver accuracy and indeed, we identify the currently best reported objective function 3.78085 as the global optimimum within our tolerance of $\epsilon = 10^-3$ with two parameters at the lower bound, approximating a switching structure with four switches. The upper bound is arbitrary to speed up the convergence and may be relaxed further at the expense of computational time.

With only four control values the problem is not as big as Problem 6.8, but inhibits some interesting convergence behavior.
Table 6.12 lists the global solution and five more local solutions plotted in Figure 6.16 and 6.17 for the given switching structure and domain. We notice quite different solution trajectories. The optimal switching times p_j are marked in green.

3.78085	5.72864	6.32495	7.33704	9.40817	10.6945

Table 6.12: Observed objective function values of local minima during the branch-and-bound for Problem (6.12).

Taking a closer look at the convergence behavior, visualized in Figure 6.18, we notice several plateaus that are not present in the convergence plots so far. They occur when branching on one the control values that enter the objective function directly and the solution is just the lower bound of this value. This happens frequently at the beginning at improves at the end.

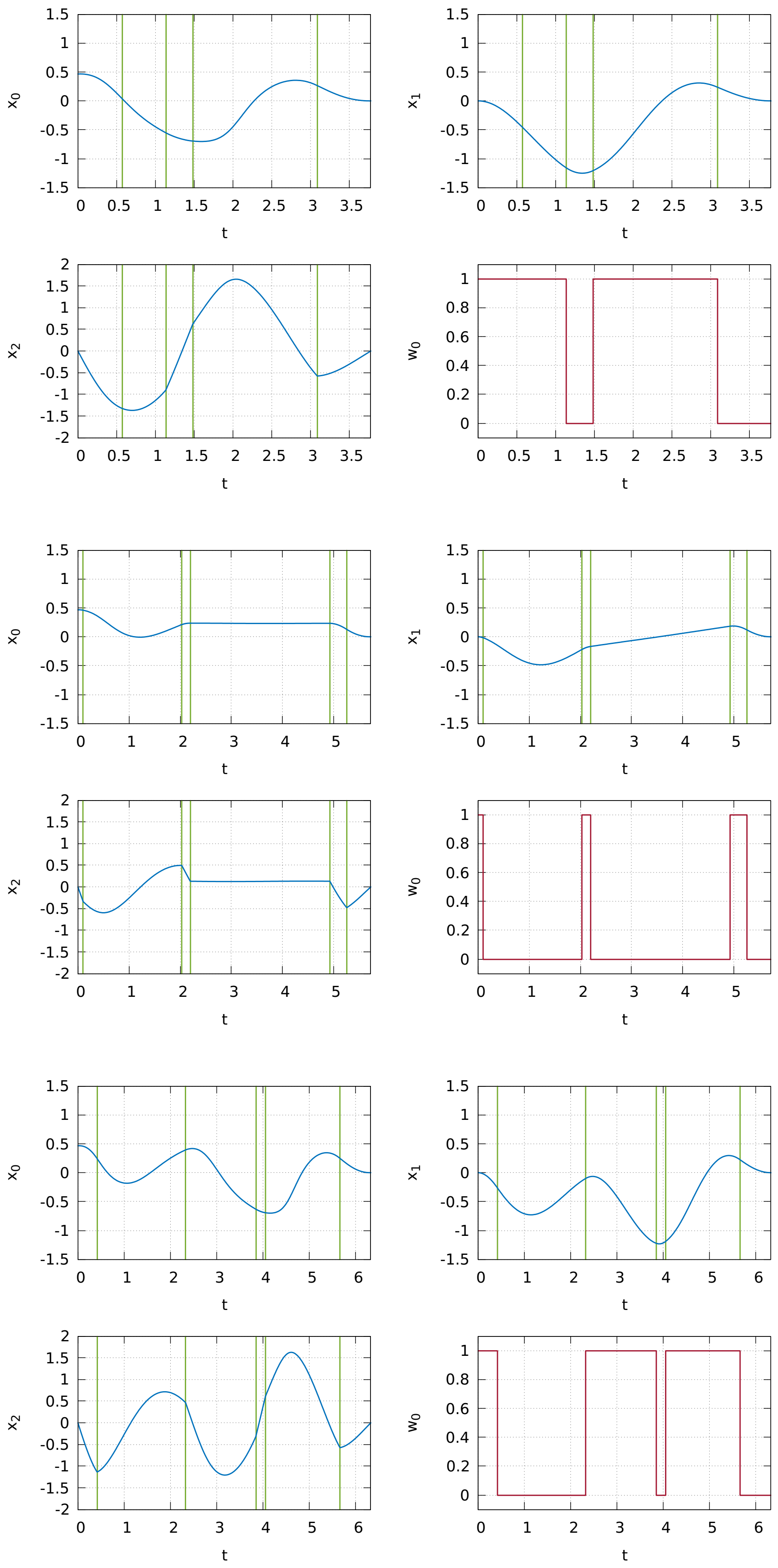


Figure 6.16: Trajectories of the global solution (top) in comparison to two worse local solutions of Problem (6.12) with six model stages and a fixed integer control. The switching times are marked in green.

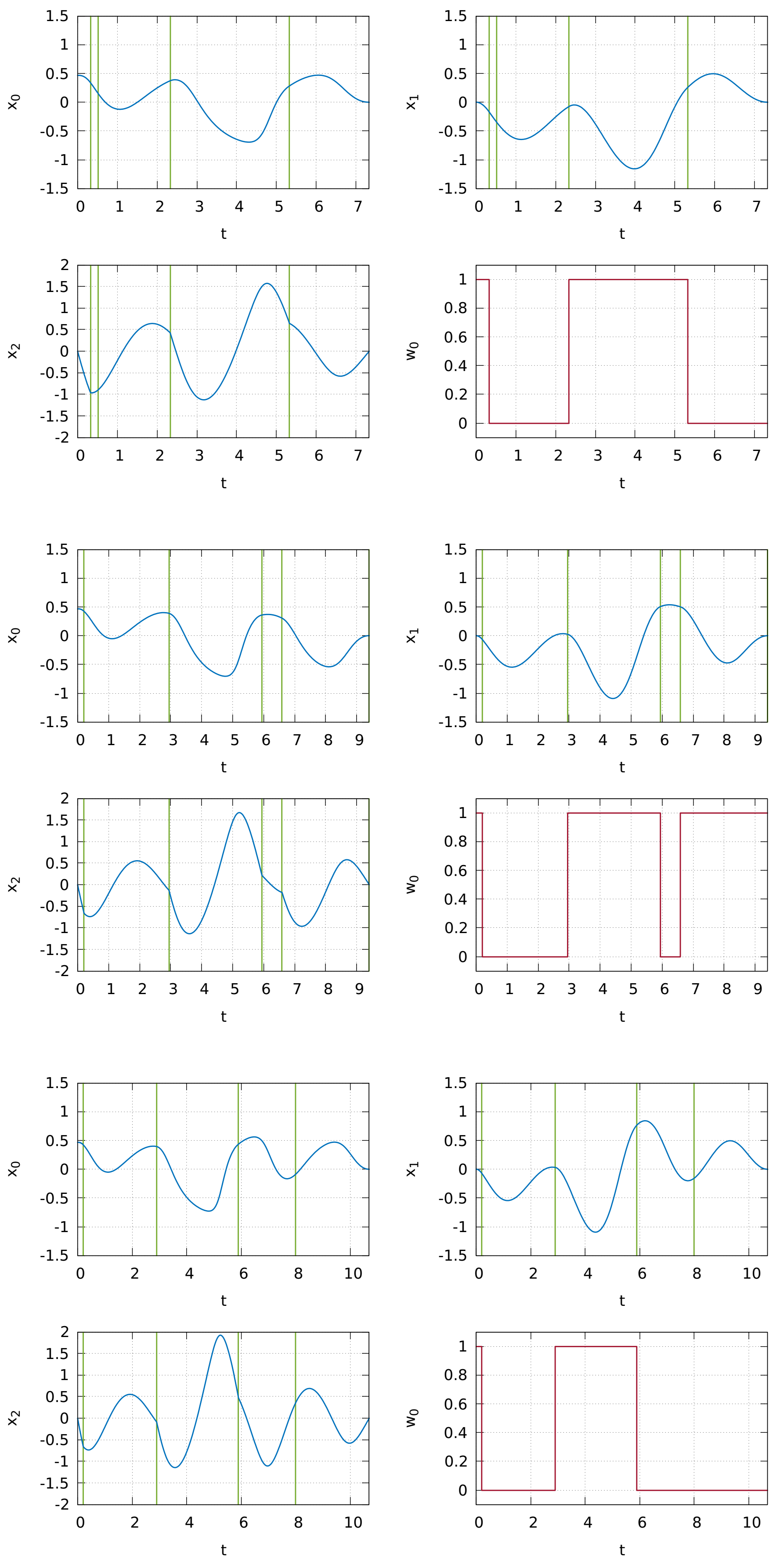


Figure 6.17: Three more local solutions and the corresponding trajectories of Problem (6.12) with six model stages and a fixed integer control. The switching times are marked in green.

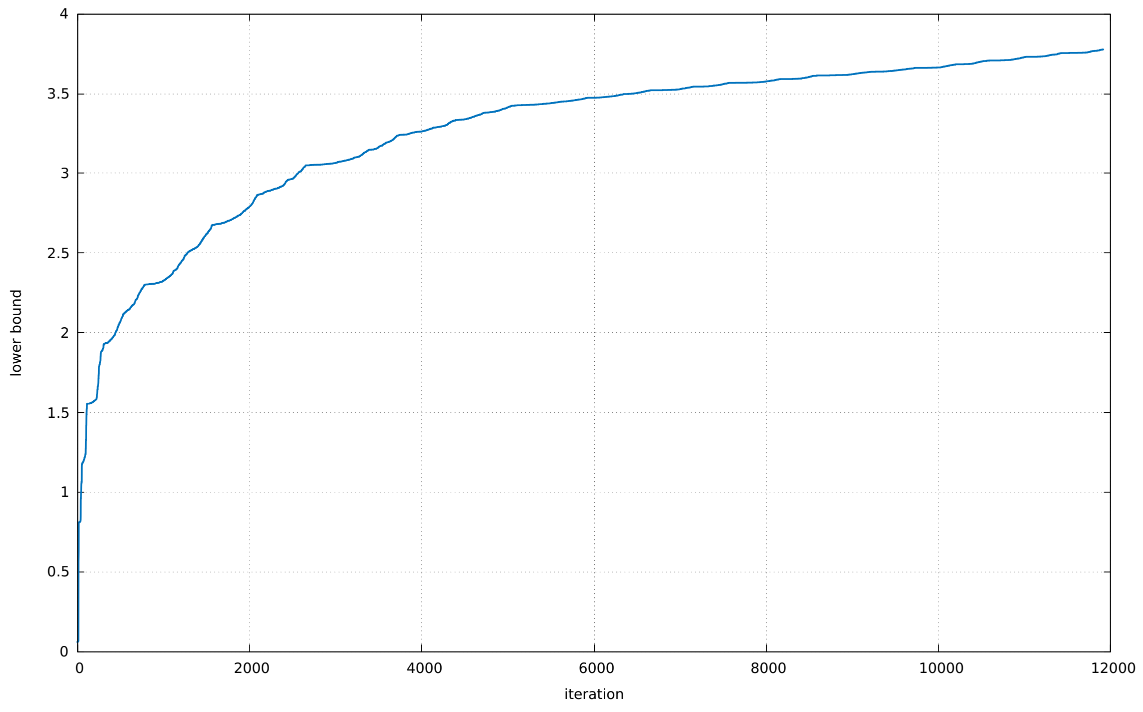


Figure 6.18: Convergence of the lower bound to the global solution for Problem (6.12) with six model stages and a fixed integer control.

6.5 Adaptively Scaled Gershgorin for NLPs

In this section, we focus on the numerical results regarding our proposed adaptively scaled Gershgorin method derived in Section 4.4.2. As the method itself is applicable for global optimization as well, we use this opportunity to showcase the flexibility of our software package GloOptCon and apply it to a number of unconstrained test cases from [Hla15]. We verify the theoretical result that the resulting convex relaxation is always at least as good and very often significantly better than either no scaling or any other fixed scaling such as using the variable interval widths.

We compare the adaptively scaled Gershgorin method for the following eight functions that are used for comparison in [Hla15] and are original from [AMF04, GF08, SWMF12].

$$\phi_0(v_0, v_1) = v_0^4 + v_1 - (v_0 + v_1^2)^2 \tag{6.13a}$$
$$\phi_1(v_0, v_1) = (1.0 + v_0 - e^{v_1})^2 \tag{6.13b}$$
$$\begin{aligned}\phi_2(v_0, v_1, v_2, v_3) =& (v_0 + 10.0v_1)^2 + 5.0(v_2 - v_3)^2 \\ &+ (v_1 - 2.0v_2)^4 + 10.0(v_0 - v_3)^4\end{aligned} \tag{6.13c}$$
$$\phi_3(v_0, v_1) = (2.0 * v_0 + v_1 - 3.0)^2 + (v_0 v_1 - 1.0)^2 \tag{6.13d}$$
$$\phi_4(v_0, v_1) = 4.0v_0^2 - 2.1v_0^4 + \frac{1.0}{3.0}v_0^6 + v_0 v_1 - 4.0v_1^2 + 4.0v_1^4 \tag{6.13e}$$
$$\begin{aligned}\phi_5(v_0, v_1, v_2) =& 100.0(v_1 - v_0^2)^2 + (1.0 - v_0)^2 + 90.0(v_3 - v_2^2)^2 + (1.0 - v_2)^2 \\ &+ 10.1((1.0 - v_1)^2 + (1.0 - v_3)^2) + 19.8((1.0 - v_1) + (1.0 - v_3))\end{aligned} \tag{6.13f}$$
$$\begin{aligned}\phi_6(v_0, v_1) =& (v_1 - \frac{5.1}{4.0\pi^2}v_0^2 + \frac{5.0}{\pi}v_0 - 6.0)^2 \\ &+ 10.0(1.0 - \frac{1.0}{8.0\pi}\cos v_0 + 10.0\end{aligned} \tag{6.13g}$$
$$\begin{aligned}\phi_7(v_0, v_1, v_2) =& 4.0v_0^2 - 2.1v_0^4 + \frac{1.0}{6.0}v_0^6 + v_0 v_1 - 4.0v_1^2 + 4.0v_1^4 \\ &+ 4.0v_1^2 - 2.1v_1^4 + \frac{1.0}{6.0}v_1^6 + v_1 v_2 - 4.0v_2^2 + 4.0v_2^4\end{aligned} \tag{6.13h}$$

with $v_0, v_1, v_2, v_3 \in \mathbb{R}$.

Table 6.13 contains the results. We notice that our convex relaxations are not only always at least as good as the results of the scaled Gershgorin method as predicted by our theoretical results in Section 4.4.2, but even equal to the results obtained with the significantly more expensive iterative bounds from [Hla15]. We note that for two-dimensional problems, the latter follows from the fact that only a single Gershgorin disc may fulfill the assumption of Lemma 22.

In higher dimensions, our specific choice of the scaling index results in an equally tight underestimator compared to the iterative method in the test cases. In comparison to the results in [Hla15], there are some differences for Φ_4 and Φ_7 which both include integer exponents and Φ_6 with a trigonometric function. The reason for this disparity is most likely that the interval library used in that paper provides a tighter evaluation of those functions.

Figure 6.19 visualized the adaptively scaled Gershgorin method for ϕ_2.

	domain	$\underline{\Phi}^{\text{std}}$	$\underline{\Phi}^{\text{sca}}$	$\underline{\Phi}^{\text{iter}}$	$\underline{\Phi}^{\text{adasca}}$
ϕ_0	$[-1.4, 5.0] \times [-1.6, 5.0]$	-2406.04	-2405.95	-2405.95	-2405.95
	$[1.0, 3.0] \times [-1.0, 1.0]$	-14.0208	-14.0208	-12.8232	-12.8232
	$[-4.0, -2.0] \times [1.0, 3.0]$	-43.6462	-43.6462	-39.4774	-39.4774
ϕ_1	$[0.0, 1.0] \times [0.0, 2.0]$	-14.4624	-12.6499	-12.6499	-12.6499
	$[-2.0, 0.0] \times [2.0, 4.0]$	0	0	24.4531	24.4531
	$[-9.0, -2.0]^2$	-0.648276	-0.648276	0.784573	0.784573
ϕ_2	$[0.0, 1.0]^4$	-85.1312	-85.1312	-80.2080	-80.2080
	$[0.0, 0.2] \times [0.0, 1.0]^3$	-41.7115	-32.5334	-27.1639	-27.1639
	$[0.0, 1.0] \times [0.0, 0.2] \times [0.0, 1.0]^2$	-75.5903	-64.9597	-64.9597	-64.9597
ϕ_3	$[0.0, 4.0]^2$	-231.046	-231.046	-231.046	-231.046
	$[1.0, 3.0] \times [-10.0, -8.0]$	136.220	136.220	142.995	142.995
	$[-1.0, 1.0] \times [7.0, 9.0]$	5.70217	5.70217	15.9564	15.9564
ϕ_4	$[-1.9, 1.9] \times [-1.1, 1.1]$	-427.382	-427.062	-427.062	-427.062
	$[-5.0, -3.0] \times [0.0, 2.0]$	101.125	101.125	101.617	101.617
	$[-2.0, 0.0] \times [-4.0, -2.0]$	5.25864	5.25864	5.75506	5.75506
ϕ_5	$[0.0, 1.0]^4$	-197.549	-197.549	-197.549	-197.549
	$[1.0, 2.0] \times [-0.5, 0.5] \times [1.0, 2.0]^2$	-86.5569	-86.5569	-75.4872	-75.4872
	$[-2.0, -0.5] \times [-2.0, -1.0] \times [-1.5, -0.5]^2$	343.800	330.345	343.107	343.107
ϕ_6	$[-5.0, 10.0] \times [0.0, 15.0]$	-1485.48	-1485.48	-1485.48	-1485.48
	$[7.0, 10.0] \times [0.0, 5.0]$	-15.1173	-16.5356	-15.1006	-15.1006
	$[-13.0, -6.0] \times [4.0, 11.0]$	99.5351	99.5351	103.859	103.859
ϕ_7	$[-2.0, 2.0]^3$	-1499.20	-1499.20	-1499.20	-1499.20
	$[2.0, 3.0]^3$	63.8959	63.8959	64.0163	64.0163
	$[-1.0, 1.0] \times [3.0, 5.0] \times [-1.0, 1.0]$	235.064	235.064	236.048	236.048

Table 6.13: Solutions of different convex relaxations of the functions $\phi_0, \ldots, \phi_7$ from Equations (6.13a) to (6.13h) over three different domains each. The column $\underline{\Phi}^{\text{std}}$ gives the lower bound obtained using an αBB relaxation based on the unscaled Gershgorin from Equation (3.20). The results from the column $\underline{\Phi}^{\text{sca}}$ are obtained using Equation (3.21) with the standard choice of $d = \overline{v} - \underline{v}$ for the scaling vector d. $\underline{\Phi}^{\text{iter}}$ is the lower bound obtained based on the iterative method described in [Hla15]. The last column $\underline{\Phi}^{\text{adasca}}$ contains the results of our efficient approach preserving the sparsity pattern of the scaled version with $d = \overline{v} - \underline{v}$.

6.6 Fast Bounds on the Second Order Sensitivities

As a proof of concept for our theoretical results from Section 4.4.4, we give results for a wrapper around `VSPODE`. Our implementation of these fast bounds on the sensitivities performs only a single integration step on the states. We calculate the intervals enclosing the remainder for the sensitivities based on the current Taylor model and the last sensitivity bounds and perform the next single integration step. A better implementation would take the sensitivity remainder during the step size control into account and perform countermeasures to the

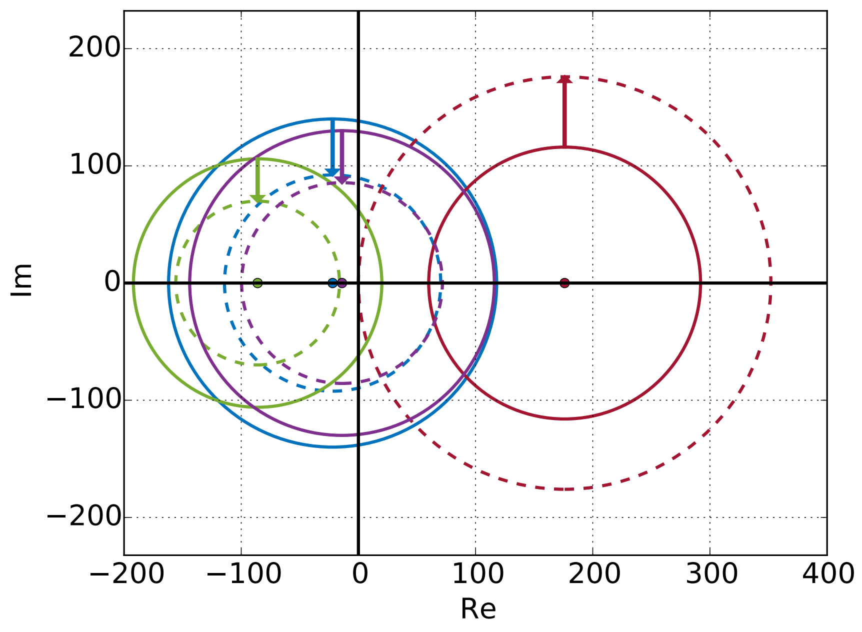


Figure 6.19: Visualization of the adaptively scaled Gershgorin method for ϕ_2.

wrapping effect on the remainder intervals. This requires a complete reimplementation of the validated integration method. Therefore, the following results are an indicator for the potential speed up for both direct single and direct multiple shooting algorithms, but not for the potential accuracy of those fast bounds.

We measure the computational time for calculating bounds for the full interval sensitivities for Problem 6.5 on the initial domain from t_0 to t_N for an increasing amount of sensitivities. To give a comparison on the overhead, we additionally give the runtime for determining the state bounds only. Furthermore, because the computational effort for states only and the fast sos method is very low, we take the median time of several runs for each multiple shooting discretization. The system size of the variational differential equations (VDE) does not increase, instead the number of integrator steps increase, because there is a forced integration stop at each multiple shooting node. The results are compiled in Table 6.14.

N	3	5	10
full VDE [s]	1.77565	3.57113	6.86278
fast sos method [s]	0.011910	0.015617	0.036301
states only [s]	0.003406	0.005124	0.030604

Table 6.14: Comparison of integrating the second-order VDEs with a validated integrator to obtain the interval second-order sensitivities with the proposal from Section 4.4.4 to differentiate the Taylor model directly (fast sos method).

Even when accounting for additional matrix operations on the remainder term to counteract the wrapping effect, the results are very promising.

6.7 Conclusion

In this chapter, we presented a numerical comparison between direct single and multiple shooting in the context of deterministic global optimization using αBB-based convex relaxations. We applied our novel multiple-shooting-based algorithm for global optimal control to a number of test problems from literature and other challenging applications where multiple local minima were observed. We verified the theoretical prediction from Chapter 4 that the relaxations are improved which in turn resulted in less iterations and a reduced computational effort. Our special branching strategy, theoretically motivated by our results in Section 4.3 and introduced in Section 4.4.1 makes in unnecessary to branch on the multiple shooting variables.
The heuristic to convexify with respect to the control functions and the control parameters only if the initial values for the states are fixed proved to be very successful in the sense that it did not converge to a suboptimal local minima while providing a speedup of several orders of magnitude on the test set.
We presented the adaptively scaled gershgorin method that provides us with fast means to scale the convex and concave relaxations that occur in the relaxed matching conditions independently on a number of nondynamic test problems from the literature.
The proposed method to obtain the necessary second-order sensitivities not by validated integration of the variational differential equation, but by modifying the validated integrator such that it uses the Taylor model directly to obtain bounds on the sensitivities showed that a significant speed up of αBB-based global optimization methods is possible using this approach.
Our software package GloOptCon was used on a wide array of test problem, including nondynamic problems, OCPs with boundary value constraints and multiple stages. The plotting capabilities gave more insight into the global optimization of optimal control problems.
To sum up, the theoretically proven increase of efficiency is verified by our numerical results. Compared to the previous direct single-shooting-based approach from the literature, the number of iterations for typical problems is more than halved. This has a positive effect on the computation time, which is reduced by almost an order of magnitude in certain instances. This in turn allows the global solution of larger optimal control problems.

Acronyms

αBB	α-Branch-and-Bound
AD	Automatic Differentiation
BB	Branch and Bound
BDF	Backward Differentiation Formula
BVP	Boundary Value Problem
DP	Dynamic Programming
END	External Numerical Differentiation
FOS	First-Order Sensitivities
GA	Genetic Algorithm
HJB	Hamilton-Jacobi-Bellman
IP	Interior Point
IND	Internal Numerical Differentiation
IVP	Initial Value Problem
MIOCP	Mixed-Integer Optimal Control Problem
NLP	Nonlinear Program
OCP	Optimal Control Problem
ODE	Ordinary Differential Equaion
PMP	Pontryagin’s Maximum Principle
QM	Quasi-Monotone
RHS	Right-Hand Side
RK	Runge-Kutta
SA	Simulated Annealing
SOS	Second-Order Sensitivities
SQP	Sequential Quadratic Programming
UML	Unified Modeling Language
VDE	Variational Differential Equation

Notation

$\mathbb{R}$	real numbers
$[\mathbb{R}]$	set of closed intervals over $\mathbb{R}$
$\underline{z}$	lower bound of z
$\overline{z}$	upper bound of z
$Z := [\underline{z}, \overline{z}]$	closed interval
$(\underline{z}, \overline{z})$	open interval
$(]/[)$	half-open interval
$w(Z)$	interval width
$m(Z)$	interval midpoint
$\lvert Z \rvert$	interval absolute value
$\frac{\mathrm{d}}{\mathrm{d}x}$	total derivative
$\frac{\partial}{\partial x}$	partial derivative
I^n	identity matrix
$0^{n \times m}$	zero matrix
$\otimes$	Kronecker product
$\succeq$	positive semi-definite
$\succ$	positive definite
$\preceq$	negative semi-definite
$\prec$	negative definite
$\bigcup$	union
$\dot{\bigcup}$	disjoint union

List of Figures

List of Tables

List of Algorithms

Bibliography

[AAF98] C.S. Adjiman, I.P. Androulakis, and C.A. Floudas. A global optimization method, αBB, for general twice-differentiable constrained NLPs – II. Implementation and computational results. *Computers & Chemical Engineering*, 22(9):1159–1179, 1998.

[AD10] J. Albersmeyer and M. Diehl. The Lifted Newton Method and its Application in Optimization. *SIAM Journal on Optimization*, 20(3):1655–1684, 2010.

[ADFN98] C.S. Adjiman, S. Dallwig, C.A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs – I. Theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998.

[AF96] C.S. Adjiman and C.A. Floudas. Rigorous convex underestimators for general twice-differentiable problems. *Journal of Global Optimization*, 9:23–40, 1996.

[Alb10] J. Albersmeyer. *Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2010.

[AMF95] I.P. Androulakis, C.D. Maranas, and C.A. Floudas. αBB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, 1995.

[AMF04] I.G. Akrotirianakis, C.A. Meyer, and C.A. Floudas. On the role of off-diagonal elements of the Hessian matrix in the construction of convex underestimators. In *Foundations of Computer Aided Process Design*, 2004.

[Bär83] V. Bär. Ein Kollokationsverfahren zur numerischen Lösung allgemeiner Mehrpunktrandwertaufgaben mit Schalt– und Sprungbedingungen mit Anwendungen in der optimalen Steuerung und der Parameteridentifizierung. Diploma thesis, Rheinische Friedrich–Wilhelms–Universität zu Bonn, 1983.

[Bar06] R. Barrio. Sensitivity Analysis of ODEs/DAEs Using the Taylor Series Method. *SIAM Journal on Scientific Computing*, 27(6):1929–1947, 2006.

[Bei12] D. Beigel. *Efficient goal-oriented global error estimation for BDF-type methods using discrete adjoints*. PhD thesis, Universität Heidelberg, 2012.

[Bel] B.M. Bell. The CppAD homepage. http://www.coin-or.org/CppAD/.

[Bel54] R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 11 1954.

[Bel57] R.E. Bellman. *Dynamic Programming*. University Press, Princeton, N.J., 6th edition, 1957. ISBN 0-486-42809-5 (paperback).

[Bel09] P. Belotti. Couenne: a user's manual. Technical report, Lehigh University, 2009.

[Bel15] B.M. Bell. CppAD: A Package for Differentiation of C++ Algorithms. Computational Infrastructure for Operations Research, 2015.

[Bie84] L.T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers & Chemical Engineering*, 8:243–248, 1984.

[BM12] A. Bompadre and A. Mitsos. Convergence rate of McCormick relaxations. *Journal of Global Optimization*, 52(1):1–28, 2012.

[Boc81] H.G. Bock. Numerische Behandlung von zustandsbeschränkten und Chebyshev-Steuerungsproblemen. Technical Report R106/81/11, Carl Cranz Gesellschaft, Heidelberg, 1981.

[Boc87] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften*. Universität Bonn, Bonn, 1987.

[BP84] H.G. Bock and K.J. Plitt. A Multiple Shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, pages 242–247, Budapest, 1984. Pergamon Press.

[BS96] C. Bendtsen and O. Stauning. FADBAD, a Flexible C++ Package for Automatic Differentiation. Technical Report IMM–REP–1996–17, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1996.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.

[Ces83] L. Cesari. *Optimization — Theory and Applications*. Springer Verlag, 1983.

[CH52] C.F. Curtiss and J.O. Hirschfelder. Integration of stiff equations. *Proc. Nat. Acad. Sci*, 38:235–243, 1952.

[CL04] B. Chachuat and M.A. Latifi. A New Approach in Deterministic Global Optimisation of Problems with Ordinary Differential Equations. In C.A. Floudas and P.M. Pardalos, editors, *Frontiers in Global Optimization*, volume 74 of *Nonconvex Optimization and Its Applications*, pages 83–108. Springer US, 2004.

[CSB05] B. Chachuat, A.B. Singer, and P.I. Barton. Global mixed-integer dynamic optimization. *AIChE Journal*, 51(8):2235–2253, 2005.

[CSB06] B. Chachuat, A.B. Singer, and P.I. Barton. Global methods for dynamic optimization and mixed-integer dynamic optimization. *Industrial and Engineering Chemistry Research*, 45(25):8573–8392, 2006.

[Die02] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik*. VDI Verlag, Düsseldorf, 2002.

[DK94] K. Du and R.B. Kearfott. The cluster problem in multivariate global optimization. *Journal of Global Optimization*, 5(3):253–265, 1994.

[EF00a] W.R. Esposito and C.A. Floudas. Deterministic Global Optimization in Nonlinear Optimal Control Problems. *Journal of Global Optimization*, 17(1–4):97–126, 2000.

[EF00b] W.R. Esposito and C.A. Floudas. Global Optimization for the Parameter Estimation of Differential-Algebraic Systems. *Industrial and Engineering Chemistry Research*, 39(5):1291–1310, 2000.

[EGK$^+$02] J. Ellson, E. Gansner, L. Koutsofios, S.C. North, and G. Woodhull. Graphviz – Open Source Graph Drawing Tools. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing*, volume 2265 of *Lecture Notes in Computer Science*, pages 483–484. Springer Berlin Heidelberg, 2002.

[Feh69] E. Fehlberg. Klassische Runge-Kutta-Formeln fünfter und siebenter Ordnung mit Schrittweiten-Kontrolle. *Computing*, 4:93–106, 1969.

[FG09] C.A. Floudas and C.E. Gounaris. A review of recent advances in global optimization. *Journal of Global Optimum*, 45(1):3–38, 2009.

[FL98] R. Fletcher and S. Leyffer. User manual for filterSQP. Technical report, University of Dundee, 1998.

[FL02] Roger Fletcher and Sven Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, 2002.

[FS69] J.E. Falk and R.M. Soland. An Algorithm for Separable Nonconvex Programming Problems. *Management Science*, 15(9):550–569, 1969.

[FSD15] J.V. Frasch, S. Sager, and M. Diehl. A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computation*, 7(3):289–329, 2015.

[FTTMB08] A. Flores-Tlacuahuac, S. Terrazas-Moreno, and L.T. Biegler. Global Optimization of Highly Nonlinear Dynamic Systems. *Industrial & Engineering Chemistry Research*, 47(8):2643–2655, 2008.

[GC94] B. Gendron and T.G. Crainic. Parallel Branch-and-Bound Algorithms: Survey and Synthesis. *Operations Research*, 42(6):1042–1066, 1994.

[Ger31] S. Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 6:749–754, 1931.

[GF08] C.E. Gounaris and C.A. Floudas. Tight convex underestimators for $\mathcal{C}^2$-continuous problems: Ii. multivariate functions. *Journal of Global Optimization*, 42(1):69–89, 2008.

[GW08] A. Griewank and A. Walther. *Evaluating Derivatives*. SIAM, second edition, 2008.

[Har02] P. Hartman. *Ordinary differential equations*, volume 38 of *Classics in Applied Mathematics*. SIAM, Philadelphia, PA, 2002. Corrected reprint of the second (1982) edition [Birkhäuser, Boston, MA; MR0658490 (83e:34002)].

[Hat14] K. Hatz. *Efficient numerical methods for hierarchical dynamic optimization with application to cerebral palsy gait modeling*. PhD thesis, Universität Heidelberg, 2014.

[HBG$^+$05a] A.C. Hindmarsh, P.N. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, and C.S. Woodward. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Transactions on Mathematical Software*, 31(3):363–396, September 2005.

[HBG$^+$05b] Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, 31(3):363–396, September 2005.

[HC14] B. Houska and B. Chachuat. Branch-and-Lift Algorithm for Deterministic Global Optimization in Nonlinear Optimal Control. *Journal of Optimization Theory and Applications*, 162(1):208–248, 2014.

[HFD11a] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[HFD11b] B. Houska, H.J. Ferreau, and M. Diehl. An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. *Automatica*, 47(10):2279–2285, 2011.

[Hla15] M. Hladík. On the efficient gerschgorin inclusion usage in the global optimization αBB method. *Journal of Global Optimization*, 61(2):235–253, 2015.

[Hol73] J.H. Holland. Genetic Algorithms and the Optimal Allocation of Trials. *SIAM Journal on Computing*, 2(2):88–105, 1973.

[HR71] G.A. Hicks and W.H. Ray. Approximation methods for optimal control systems. *Can. J. Chem. Engng.*, 49:522–528, 1971.

[HT96] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer Berlin Heidelberg, 1996.

[HVC13] B. Houska, M.E. Villanueva, and B. Chachuat. A validated integration algorithm for nonlinear ODEs using Taylor models and ellipsoidal calculus. In *IEEE 52nd Annual Conference on Decision and Control (CDC)*, pages 484–489, Dec 2013.

[Kam32] E. Kamke. Zur Theorie der Systeme gewöhnlicher differentialgleichungen. II. *Acta Mathematica*, 58(1):57–85, 1932.

[Kam15] A. Kaminsky. *BIG CPU, BIG DATA – Solving the World's Toughest Computational Problems with Parallel Computing*. 2015.

[KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[Kha02] H.K. Khalil. *Nonlinear Systems*. Prentice Hall, 2002.

[Kir10] C. Kirches. *Fast numerical methods for mixed-integer nonlinear model-predictive control*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, July 2010. Available at http://www.ub.uni-heidelberg.de/archiv/11636/.

[KN03] C.Y. Kaya and J.L. Noakes. A Computational Method for Time-Optimal Control. *Journal of Optimization Theory and Applications*, 117:69–92, 2003.

[Knü94] O. Knüppel. PROFIL/BIAS – A fast interval library. *Computing*, 53(3–4), 1994.

[KPBS12] S. Körkel, A. Potschka, H.G. Bock, and S. Sager. A Multiple Shooting Formulation for Optimum Experimental Design. *Mathematical Programming*, 2012. (submitted revisions).

[LD60] A.H. Land and A.G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[Lei99] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[Len14] S. M. Lenz. *Impulsive Hybrid Discrete-Continuous Delay Differential Equations*. PhD thesis, Universität Heidelberg, 2014.

[LMSK63] J.D.C. Little, K.G. Murty, D.W. Sweeney, and C. Karel. An Algorithm for the Traveling Salesman Problem. *Operations Research*, 11(6):972–989, 1963.

[Loh92] R.J. Lohner. Computation of Guaranteed Enclosures for the Solutions of Ordinary Initial and Boundary Value Problems. In J.R. Cash and I. Gladwell, editors, *Computational Ordinary Differential Equations*, pages 425–436. Clarendon Press, 1992.

[LS07a] Y. Lin and M.A. Stadtherr. Deterministic global optimization of nonlinear dynamic systems. *AIChE Journal*, 53(4):866–875, 2007.

[LS07b] Y. Lin and M.A. Stadtherr. Validated solutions of initial value problems for parametric ODEs. *Applied Numerical Mathematics*, 57(10):1145–1162, Oct 2007.

[LTW$^+$06] M. Lerch, G. Tischler, J. Wolff von Gudenberg, W. Hofschuster, and W. Krämer. FILIB++, a Fast Interval Library Supporting Containment Computations. *ACM Transactions on Mathematical Software*, 32(2):299–324, June 2006.

[Luu90] R. Luus. Optimal control by dynamic programming using systematic reduction in grid size. *International Journal of Control*, 51(5):995–1013, 1990.

[MCB09] A. Mitsos, B. Chachuat, and P.I. Barton. McCormick-Based Relaxations of Algorithms. *SIAM Journal on Optimization*, 20(2):573–601, 2009.

[McC76] G.P. McCormick. Computability of global solutions to factorable nonconvex programs. Part I. Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[MF94] C.D. Maranas and C.A. Floudas. Global minimum potential energy conformations of small molecules. *Journal of Global Optimization*, 4(2):135–170, 1994.

[MKC09] R.E. Moore, R.B. Kearfott, and M.J. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[Moo66] R.E. Moore. *Interval analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[Müller27] M. Müller. Über das Fundamentaltheorem in der Theorie der gewöhnlichen differentialgleichungen. *Mathematische Zeitschrift*, 26(1):619–645, 1927.

[Ned06] N.S. Nedialkov. Interval Tools for ODEs and DAEs. In *12th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN)*, pages 4–4, Sept 2006.

[Neu04] A. Neumaier. *Complete Search in Continuous Global Optimization and Constraint Satisfaction*, pages 271–369. Cambridge University Press, 2004.

[NJP01] N.S. Nedialkov, K.R. Jackson, and J.D. Pryce. An Effective High-Order Interval Method for Validating Existence and Uniqueness of the Solution of an IVP for an ODE. *Reliable Computing*, 7(6):449–465, 2001.

[NW06] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Verlag, Berlin Heidelberg New York, second edition, 2006. ISBN 0-387-30303-0 (hardcover).

[OB05] D.B. Özyurt and P.I. Barton. Cheap Second Order Directional Derivatives of Stiff ODE Embedded Functionals. *SIAM Journal on Scientific Computing*, 26(5):1725–1743, 2005.

[Ope13] OpenMP Architecture Review Board. OpenMP Application Program Interface Version 4.0, July 2013.

[PA02] I. Papamichail and C.S. Adjiman. A Rigorous Global Optimization Algorithm for Problems with Ordinary Differential Equations. *Journal of Global Optimization*, 24(1):1–33, 2002.

[PA04] I. Papamichail and C.S. Adjiman. Global optimization of dynamic systems. *Computers & Chemical Engineering*, 28:403–415, 2004.

[PA05] I. Papamichail and C.S. Adjiman. Proof of Convergence for a Global Optimization Algorithm for Problems with Ordinary Differential Equations. *Journal of Global Optimization*, 33:83–107, 2005.

[PBGM62] L.S. Pontryagin, V.G. Boltyanski, R.V. Gamkrelidze, and E.F. Miscenko. *The Mathematical Theory of Optimal Processes*. Wiley, Chichester, 1962.

[Pli81] K.J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Diploma thesis, Rheinische Friedrich–Wilhelms–Universität Bonn, 1981.

[Pot06] A. Potschka. Handling Path Constraints in a Direct Multiple Shooting Method for Optimal Control Problems. Diploma thesis, Universität Heidelberg, 2006.

[Pot11] A. Potschka. *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints*. PhD thesis, Universität Heidelberg, 2011.

[Rih94] R. Rihm. Interval methods for initial value problems in ODEs. In J. Herzberger, editor, *Topics in Validated Computations*, Studies in computational mathematics, pages 173–207. Elsevier, 1994.

[RS72] R.D. Russell and L.F. Shampine. A collocation method for boundary value problems. *Numerische Mathematik*, 19(1):1–28, 1972.

[Sag] S. Sager. MIOCP benchmark site. http://mintoc.de.

[Sag05] S. Sager. *Numerical methods for mixed–integer optimal control problems*. Der andere Verlag, Tönning, Lübeck, Marburg, 2005. ISBN 3-89959-416-9.

[Sag06] S. Sager. *Numerical methods for mixed–integer optimal control problems*. PhD thesis, Universität Heidelberg, 2006.

[Sag12] S. Sager. A benchmark library of mixed-integer optimal control problems. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 631–670. Springer, 2012.

[Sah13] N.V. Sahinidis. *BARON 12.1.0: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual*, 2013.

[SB06] A.B. Singer and P.I. Barton. Global Optimization with Nonlinear Ordinary Differential Equations. *Journal of Global Optimization*, 34(2):159–190, 2006.

[SBD$^+$11] S. Sager, C. Barth, H. Diedam, M. Engelhart, and J. Funke. Optimization as an Analysis Tool for Human Complex Problem Solving. *SIAM Journal on Optimization*, 21(3):936–959, 2011.

[Sch11] D. Scholz. *Deterministic Global Optimization: Geometric Branch-and-bound Methods and their Applications*. Springer Optimization and Its Applications. Springer, 2011.

[SCM15] S. Sager, M. Claeys, and F. Messine. Efficient upper and lower bounds for global mixed-integer optimal control. *Journal of Global Optimization*, 61(4):721–743, 2015.

[Sco12] J.K. Scott. *Reachability analysis and deterministic global optimization of differential-algebraic systems*. PhD thesis, Massachusetts Institute of Technology, June 2012.

[SEG$^+$13] M. Schlueter, S.O. Erb, M. Gerdts, S. Kemble, and J.-J. Rückmann. MIDACO on MINLP space applications. *Advances in Space Research*, 51(7):1116–1131, 2013.

[Sha05] L.F. Shampine. Erros estimation and control for ODEs. *J. Sci. Comp.*, 25(1):3–16, 2005.

[Sin04] A.B. Singer. *Global Dynamic Optimization*. PhD thesis, Massachusetts Institute of Technology, 2004.

[SKS13] O. Stein, P. Kirst, and P. Steuermann. An Enhanced Spatial Branch-and-Bound Method in Global Optimization with Nonconvex Constraints. *Optimization Online*, 2013.

[SN04] H. Schichl and A. Neumaier. Exclusion Regions for Systems of Equations. *SIAM Journal on Numerical Analysis*, 42(1):383–408, 2004.

[SRB09] S. Sager, G. Reinelt, and H.G. Bock. Direct Methods With Maximal Lower Bound for Mixed-Integer Optimal Control Problems. *Mathematical Programming*, 118(1):109–149, 2009.

[SS78] R.W.H. Sargent and G.R. Sullivan. The development of an efficient optimal control package. In J. Stoer, editor, *Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2*, Heidelberg, 1978. Springer.

[SSB11] J.K. Scott, M.D. Stuber, and P.I. Barton. Generalized McCormick relaxations. *Journal of Global Optimization*, 51(4):569–606, 2011.

[SW14] Anders Skjäl and Tapio Westerlund. New methods for calculating αBB-type underestimators. *Journal of Global Optimization*, 58(3):411–427, 2014.

[SWMF12] A. Skjäl, T. Westerlund, R. Misener, and C.A. Floudas. A Generalization of the Classical αBB Convex Underestimation via Diagonal and Nondiagonal Quadratic Terms. *Journal of Optimization Theory and Applications*, 154(2):462–490, 2012.

[TH88] H. Tuy and R. Horst. Convergence and restart in branch-and-bound algorithms for global optimization. Application to concave minimization and D.C. Optimization problems. *Mathematical Programming*, 41(1–3):161–183, 1988.

[TS05] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.

[VBCB99] V.S. Vassiliadis, E. Balsa-Canto, and J.R. Banga. Second-order sensitivities of general dynamic systems with application to optimal control problems. *Chemical Engineering Science*, 54(17):3851–3860, 1999.

[Ved11] M. Vedder. Optimierung von Roboterbahnen unter Verwendung des Pontryaginschen Maximumprinzips am Beispiel eines 2D-Knickarmroboters. Master's thesis, Universität Heidelberg, 2011.

[VHC15] M.E. Villanueva, B. Houska, and B. Chachuat. Unified framework for the propagation of continuous-time enclosures for parametric nonlinear ODEs. *Journal of Global Optimization*, 62(3):575–613, 2015.

[Wal64] W. Walter. *Differential- und Integral-Ungleichungen und ihre Anwendung bei Abschätzungs- und Eindeutigkeitsproblemen*, volume 2 of *Springer Tracts in Natural Philosophy*. Springer-Verlag Berlin Heidelberg GmbH, 1 edition, 1964.

[Wal71] W. Walter. Ordinary differential inequalities in ordered Banach spaces. *Journal of Differential Equations*, 9(2):253–261, 1971.

[WB06] A. Wächter and L.T. Biegler. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1):25–57, 2006.

[Wes12] T. Westerlund. A reformulation framework for global optimization. 21st International Symposium on Mathematical Programming (ISMP), 2012.

[WSB14] A. Wechsung, S.D. Schaber, and P.I. Barton. The cluster problem revisited. *Journal of Global Optimization*, 58(3):429–438, 2014.

[WY77] C.S. Wen and T.F. Yen. Optimization of oil shale pyrolysis. *Chemical Engineering Science*, 32(3):346–349, 1977.